Berliner Informatik-Tage

bit '85

Humboldt-Universität zu Berlin

Tagungsbericht Band 2

SEKTION 2
Bildungsaspekte

Conrad, Elmar
Kade, Gisela
Schröder, Helmut
Seidel, Helmar

Einordnung und Softwareunterstützung von interaktiven Arbeitsweisen im Terminalkabinett innerhalb der Grundausbildung Informationsverarbeitung

1. Ausgangspunkt und Zielbestimmung

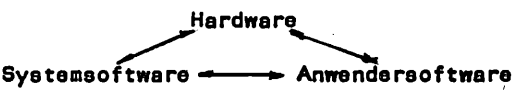
Für das Lehrgebiet Informationsverarbeitung (IV) wird im Lehrprogramm für die Ausbildung in Technischen Wissenschaften als wesentliche Zielstellung formuliert:
"Der Student wird befähigt, für die Lösung einfacher Aufgaben seines Fachgebietes Algorithmen selbständig aufzustellen, mittels einer problemorientierten Sprache daraus ein funktionsfähiges Programm zu formulieren, dieses unter Nutzung der Hilfsmittel eines leistungsfähigen Betriebssystems zu testen und die bei der Abarbeitung desselben erzielten Ergebnisse zu analysieren."

Die geforderte Selbständigkeit im Umgang mit einem leistungsfähigen Betriebssystem konnte durch den früher praktizierten closed-job-Betrieb nur teilweise erreicht werden. Auch das danach an der Ingenieurhochschule Zwickau am KRS 4201 durchgeführte Praktikum brachte für den Lehrabschnitt Betriebssysteme kaum eine effektive Untermauerung, jedoch wurde durch den unmittelbaren Rechnerkontakt eine Motivierung der Studenten erreicht.
Ziel der interaktiven Arbeitsweise ist es, die Studenten neben den erforderlichen theoretischen Kenntnissen mit ausreichenden praktischen Erfahrungen auszustatten und sie zu motivieren, selbständig im weiteren Studienverlauf und in der Praxis das Rationalisierungsmittel Rechentechnik einzusetzen.
Dieses Ziel kann nur erreicht werden, wenn die IV-Grundausbildung im Studium so zeitig wie möglich beginnt und über den gesamten Verlauf dieser Ausbildung durch häufigen Rechnerkontakt eine routinemäßige Handhabung der Rechentechnik erreicht wird.

Eine Erziehung zu Gewissenhaftigkeit, Zielstrebigkeit und Ausdauer wird durch die auftretenden Erfolgserlebnisse unterstützt.

2. Einordnung in die Seminarführung

Für einen frühzeitigen Rechnerkontakt sind im Prinzip Kenntnisse zur Hard- und Software des Rechners erforderlich.
Als Strukturträger der Grundausbildung Informationsverarbeitung können



angegeben werden.
Die Entwicklung von Softwarelösungen bildet den Schwerpunkt der Ausbildung. Es geht dabei nicht nur um eine Algorithmierung und Programmierung schlechthin, sondern um die Lehre einer Problemlösungsmethodik. Diese Methodik wird in Vorlesungen und Seminaren angewendet. Folgende Entwurfshilfsmittel werden genutzt:

- Strukturübersicht und Übersichtsdiagramme von HIPO
- strukturierter Leitlinien-PAP mit den Strukturelementen Sequenz, Alternative und Zyklus
- strukturierte Programmierung
- Programmentwicklung im Dialog.

Im folgenden Abschnitt geht es um die Frage, wie die ersten Terminalsitzungen in die Seminarführung integriert werden.
Das dritte Seminar wird im Rechentechnischen Kabinett durchgeführt. Zur Vorbereitung dieser ersten Terminalsitzung stehen in der Regel zwei Vorlesungen und zwei Seminare zur Verfügung.
Diese werden genutzt, um neben einer kurzen Einführung den prinzipiellen Aufbau und die Arbeitsweise einer EDVA, die Problemlösungsschritte bei der Erarbeitung eines EDV-Projektes und die Komponenten eines Betriebssystems grob zu skizzieren. Innerhalb der zweiten Vorlesung wird an-

hand eines linearen FORTRAN-Programmes mit der problemorientierten Programmierung begonnen.
Die ersten beiden Seminare dienen dazu, an einem linearen Programm alle Problemlösungsschritte zu absolvieren bzw. zu erläutern. Als Darstellungsmittel werden das HIPO-Übersichtsdiagramm und der strukturierte PAP genutzt. Anhand eines groben Datenflußplanes werden den Studenten die Technologie der Programmentwicklung im Dialog vorgestellt und das Dateikonzept erläutert.
Ein Arbeitsmaterial wird zur Vorbereitung auf das dritte Seminar ausgehändigt. Es enthält nur die wesentlichsten Kommandos und Schalter zu MCR, PIP, EDI, FOR und TKB.
Das Ziel der ersten Terminalsitzung ist es, die Studenten zu befähigen, selbständig den Rechner zur Quelltexteingabe und -korrektur sowie zur Arbeit mit dem Anwendersystem zu befähigen.
Es werden folgende Kommandos bzw. Dienstprogramme genutzt: HEL, FLI, EDI, PIP, STA, @ RETTEN und BYE. Den Schwerpunkt bildet die Nutzung des Editors (EDI) zur Korrektur einer vorgefertigten Textdatei. Nach diesem Seminar sind im Selbststudium die Anwendersystemkomponenten zur Einführung in FORTRAN zu nutzen. Außerdem kann der Student mit Hilfe eines "Leistungskontrollprogramms" zu arithmetischen Ausdrücken sein Wissen überprüfen. Ebenso ist der Tischrechnermodus (BAS) nutzbar.
Das vierte und fünfte Seminar werden eingesetzt, um die Problemlösungsschritte bis einschließlich der Programmierung an linearen und verzweigten Programmen zu üben. Die dritte Vorlesung schafft dazu die Voraussetzungen.
Im sechsten Seminar werden anhand des Datenflußplanes aus dem zweiten Seminar alle Schritte der Programmentwicklung im Dialog am Rechner von den Studenten absolviert. Dabei werden genutzt: EDI, FOR bzw. @ FELI, TKB und RUN.
Danach hat jeder Student an einem einfachen Beispiel alle Schritte der Problemlösung einschließlich des Rechnertest und der Analyse der Ergebnisse selbst durchgeführt.
Ausgehend von dem dadurch gewonnenen Verständis für den

Problemlösungsprozeß, werden die einzelnen Problemlösungsschritte in den folgenden Lehrveranstaltungen näher erläutert und an Beispielen geübt.

3. Software für die Unterstützung interaktiver Arbeitsweisen

3.1. Ausgangssituation und Zielstellung

Bei Aufnahme der Arbeiten im Jahre 1983 als Einsatzvorbereitung für den Nachfolgerechner des R300 (K 1630 oder CM 4-20) stand nachnutzbare Software, die auch Möglichkeiten von RGU (CAI) beinhaltet, nicht zur Verfügung. Entwicklungen für SKR waren nicht geplant und der für ESER geplante Import von RGU-Software aus der UdSSR war für uns nicht relevant. Die Qualifizierung unserer Ausbildung verlangte jedoch einen schnellen Einsatz ausbildungsspezifischer Software. Die Entwicklungsarbeiten erfolgten auf einem K 1620. Mit Installation einer CM 4-20 im Jahre 1984 wurde die entwickelte Software auf diese übernommen und mit Beginn des Studienjahres 1984/85 in der Lehre eingesetzt.

Die Arbeiten an einem terminalorientierten interaktiven Praktikumssystem (TIPS) zielten zunächst auf eine Unterstützung des Lernenden und Lehrenden bei der Durchführung von Praktika. Die rege Nutzung des Rechners im Selbststudium und die damit verbundenen Probleme für den lernenden Nutzer führten zur Zielstellung, diesen durch ein Anwendersystem /1/ zu unterstützen. D. h. in diesem Anwendersystem müssen Auskunfts- und Informationsmodus, Programmentwicklungsmodus, Tischerrechnermodus und die Abarbeitung lauffähiger Objekte enthalten sein.

Der Informationsmodus ist so konzipiert, daß mit ihm auch rechnergestütztes Lernen, Abarbeitung von Leistungskontrollen und Erstellung von sequentiellen Dateien erfolgen kann.

Eine Individualisierung der Dialogführung soll durch die wahlweise Nutzung der Menü-Technik oder von Kommandos

möglich sein. Das System wurde modular konzipiert und bei der Realisierung einzelner Komponenten achteten wir auf ihre Nachnutzbarkeit.

3.2. Unterstützung der Arbeit des Nutzers (des Studenten) durch das Programmpaket DIRAS /2/

Der selbständig arbeitende, jedoch durch das Anwendersystem geführte Student wird durch das den Dialog rechnerseitig führende Programm DIRAS in seiner Arbeit unterstützt. Dabei entnimmt das vom Unterrichtsgegenstand unabhängige Programm DIRAS Textinformationen aus einer Direktzugriffsdatei und gibt diese auf Bildschirm aus.
Eine solche auszugebende Einheit von Textinformationen, deren Empfang vom Teilnehmer mit einer Eingabe bzw. Tastendruck quittiert werden muß, wird nachfolgend Bild genannt. Intern ist die Gesamtheit aller Bilder mit gemeinsamen Sachbezug in einer Direktzugriffsdatei abgespeichert. Eine solche Direktzugriffsdatei ist also die interne Form eines Lehrprogrammes.
Da sich Bildfolgen, die die Arbeit des Teilnehmers lediglich lenken, in ihrer internen Darstellung nicht von echten Lehrprogrammen unterscheiden, werden sie nachfolgend etwas ungenau ebenfalls als Lehrprogramm bezeichnet. Vom Start von DIRAS an bis zur Beendigung des Programmlaufs kann der Teilnehmer nacheinander in mehreren Lehrprogrammen arbeiten. Durch die gegebene oder unterlassene Antwort entscheidet sich der Teilnehmer für genau eine aus maximal 13 verschiedenen Fortsetzungsmöglichkeiten, d. h. die vom Teilnehmer gegebene Antwort (die in Kurzform gegeben werden darf), wird verglichen mit maximal 13 zu jedem Bild angebbaren Antwortvorgaben des Autors. Dabei kann der Autor folgende Fortsetzungsarten als Folge einer bestimmten Antwort veranlassen:

- Wechsel zu einem bestimmten Bild desselben Lehrprogrammes

- Nochmaliges Anzeigen des gleichen Bildes (z. B. nach unsinnigen Antworten)
- Wechsel zu einem von drei Nachfolgebildern in Abhängigkeit von den bisher im Sitzungsverlauf vom Nutzer erwiesenen Fähigkeiten
- Ausgabe einer Bewertung der Arbeit des Nutzers und Fortsetzung mit einem bestimmten Bild desselben Lehrprogrammes
- Übergang in die PAUSE-Anweisung, wodurch zwischenzeitliche Arbeit des Nutzers in anderen Programmen möglich wird
- Wechsel zu einem bestimmten Bild eines anderen Lehrprogrammes (Schließen der Direktzugriffsdatei und Eröffnen einer anderen Direktzugriffsdatei)

Während der Nutzer entsprechend dem vom Autor festgesetzten Menü antwortet, werden implizit für den Nutzer die dargestellten Handlungen ausgeführt. Die eingangs dargestellten Arbeitsmodi des Anwendersystems werden durch DIRAS auf folgende Weise realisiert:

- der Informationsmodus
  durch bloße Dialogführung mittels einer oder mehrerer Direktzugriffsdateien
- der Leistungskontrollmodus
  durch Dialogführung und Registrierung des Erreichens bestimmter Bilder.
  Diese Kennzeichnungen werden klassifiziert: eine Klasse "richtig", mehrere Klassen "falsch"
- der Dateierstellungsmodus
  Dialogführung , wobei manche Antworten des Teilnehmers in einer sequentiellen Datei gesammelt werden.
  Diese kann später als Eingabedatei eines anderen Programmes verwendet werden

- der Tischrechnermodus

  Er wird lediglich unterstützt: Die benötigten Kommandos für den BASIC-Interpreter werden dem Teilnehmer vorgestellt, danach geht DIRAS in eine PAUSE über. Nach erfolgter Arbeit mit BASIC wird die Arbeit im Programm DIRAS wieder aufgenommen.

- der Programmentwicklungsmodus

  Er wird lediglich unterstützt: Der Werdegang eines FORTRAN-Programmes wird vorgestellt. Erforderliche Kommandos werden auf Bildschirm ausgegeben. Jeweils nach der Darstellung eines Kommandos geht DIRAS in eine PAUSE-Anweisung über. Das dargestellte Kommando kann sofort, d. h. solange das Bild noch sichtbar ist, in die Rechenanlage eingegeben werden. Durch den Dialog wird der Student bis zum Start des arbeitsfähigen Programmes geführt.

- Nutzung anderer Programme und Dienstprogramme (Operatoren)

  Die Arbeitsweise ist hier die gleiche, die Anleitung erfolgt über ein DIRAS-Lehrprogramm, nach PAUSE oder Stop können die Handlungen vom Teilnehmer ausgeführt werden.

Wie sieht nun die Arbeit des Teilnehmers im Anwendersystem aus?

Durch Eingabe von START wird das unter diesem Namen installierte Programm DIRAS gestartet. Es werden einige persönliche Angaben des Teilnehmers (Name, Seminargruppe) angefordert. Danach wird ein bestimmtes Lehrprogramm angesteuert, in dem auf weitere Programme verwiesen wird.
Die Einzelbilder erscheinen je nach Maßgabe des Autors vom oberen Rand des Bildschirmes an oder als Fortsetzung eines bereits sichtbaren Bildes. Bei ungültigen Antworten erscheint ein vom Autor für diesen Fall zugeordnetes Nachfolgebild oder dasselbe Bild noch einmal.

In vom Autor festgelegten Fällen können die Antworten als Zahl gedeutet werden.
Die Eingabe muß in einer üblichen Darstellung erfolgen. Der Teilnehmer ist an keinen Zahlentyp gebunden. Solche Antworten werden bezüglich der Relationen > , = oder < mit den vom Autor notierten Vorgaben verglichen (Beispiel: hat der Autor ## > 1Ø1 notiert, gelten alle Teilnehmerantworten richtig, deren Wert größer als 1Ø1 ist). Für nicht interpretierbare Zahlendarstellungen des Teilnehmers werden spezielle Fehlermeldungen erzeugt, die Antwort gilt dann ihrem Werte nach als falsch. Warnungen weisen darauf hin, daß "bedenkliche" Darstellungen vom Programm abgeändert wurden. Auch bei unsinnigen Antworten ist kein "Absturz" des Programmes zu befürchten.

Unabhängig von den vom Autor vorgesehenen Antworten kann der Teilnehmer an beliebiger Stelle durch Eingabe reservierter Antworten folgende Handlungen erreichen:

- Rückverfolgung des Dialogs in einem Lehrprogramm
  es können alle bisher gesehenen Bilder des Lehrprogramms und das Anfangsbild des Lehrprogrammes gerufen werden, in dem der Teilnehmer gerade arbeitet
- Abbruch des Dialogs
- Einleitung der Direktanwahl von Bildern
  Es können maximal 40 Bilder jedes Lehrprogrammes mit Namen versehen werden.
  Unabhängig vom bisher beschrittenen Dialog lassen sich diese Bilder vom Teilnehmer direkt anwählen, ohne daß er das gesamte dorthin führende Frage-Antwort-Spiel durchführen muß.
  Dieser Prozeß ist vom Autor durch Bekanntgabe seiner Verzeichnisse als Bildfolge zu unterstützen.

Die drei letztgenannten Möglichkeiten existieren im Dateierstellungsmodus und im Leistungskontrollmodus nicht, damit z. B. nach Erläuterung der Aufgabenlösung Teile der Leistungskontrollen nicht noch einmal durchlaufen werden können. Durch das Leistungskontrollabschlußbild (Bewertung) werden die Einschränkungen wieder aufgehoben.

Um ein Lehrprogramm für DIRAS zu erzeugen, muß der Lehrer (Autor) die einzelnen Bilder umgangssprachlich notieren. Die Notation ist durch Kopfzeilen zu ergänzen. In diesen stehen Bildnummer, bei direkt anwählbaren Bildern der Bildname, die Antwortvorgaben und die zugehörigen Bildnummern der Nachfolgebilder.
Die Notation wird von einem Programm übernommen und in die Direktzugriffsdatei eingegliedert. Änderung von Bildtexten geschieht durch Neueintragung des geänderten Textes. Ein Reorganisationsprogramm sorgt dafür, daß nicht mehr gültige Bildtexte in der Datei durch neu hinzukommende Bildtexte überschrieben werden.

das Programmpaket DPG /3/

DPG wurde in rd 10/84 bereits beschrieben. Mit einem automatisch generierten Programm von DPG kann der Teilnehmer genauso arbeiten wie mit einem Lehrprogramm im System DIRAS. Da aus einem Lehrprogramm von DPG immer eine Task erzeugt wird, ist ein Wechsel von einem Lehrprogramm in ein anderes jedoch nicht möglich.
Automatisch mit DPG erzeugte Dialogunterprogramme können mit handgeschriebenen Verarbeitungsmoduln zu arbeitsfähigen Programmen verbunden werden. Die Eingabe von Zahlenwerten an COMMONvariable ist durch den Teilnehmer am Ende jedes beliebigen Bildes möglich.
DPG besteht aus einem Bildunterprogrammgenerator, einem Hauptprogrammgenerator und einem Generator für die zugehörige Oberlagerungsbeschreibungsdatei sowie aus verschiedenen Objektmoduln. Der Dialogteil ist ähnlich wie bei DIRAS umgangsprachlich zu notieren.

das Programm FELI /4/

Das für die Arbeit im Rechentechnischen Kabinett eingesetzte Kommandoprogramm FELI, welches mit einer Task gleichen Namens zusammenarbeitet, bereitet die vom FORTRAN-Compiler FOR erzeugten Übersetzungsprotokolle für die Ausgabe auf

Bildschirm vor und gibt diese auf Bildschirm aus.
Dabei werden die Fehlermeldungen des Compilers durch deutschsprachige Erläuterungen bzw. Vermutungen über die Fehlerursache ergänzt, so daß der Student die dargestellten Sachverhalte an den sichtbaren Programmzeilen überprüfen kann.
Fehlermeldungen, die in den Übersetzungsprotokollen erst unter den Programmtexten erscheinen (mit Verweis auf die Zeile, in der der Fehler bemerkt wird), werden in den Text an die entsprechende Zeile eingestreut, so daß fehlerhafte Zeile und Fehlermeldung gleichzeitig auf Bildschirm sichtbar sind. Das Anschauen der Programme am Bildschirm der CM 4-20 wird durch das Dienstprogramm FLI erleichtert, welches das Rollen des Textes auf dem Bildschirm verhindert.

4. Erste Erfahrungen

Die Studenten sind bis auf wenige Ausnahmen motiviert durch die sie interessierende neue Technik und durch die Kenntnis einiger sie im späteren Studium erwartender Aufgabenstellungen (z. B. in der Sektion Kraftfahrzeugtechnik die Anwendung des Systems PLOT (Grafik-System) zum automatisierten Konstruieren und Nutzung der Rechentechnik zur Wellenberechnung oder in der Sektion Technologie die Arbeit am rechnergestützten Technologenarbeitsplatz und die Nutzung des sektionseigenen Prozeßrechners K 1630 in den höheren Semestern). Die gestellten Selbststudienaufgaben und eine häufig zu beobachtende geweckte schöpferische Neugierde haben eine hohe Auslastung des Rechentechnischen Kabinetts in der 2. und 3. Schicht zur Folge. Engpaß ist die begrenzte Verfügbarkeit der Rechentechnik. Wegen der z. T. ungenügenden Vorbereitung der einzelnen Sitzungen durch die Studenten verschlechtert sich diese Situation noch etwas. Deshalb ist zu Beginn der selbständigen Nutzung eine Konsultationsmöglichkeit effektiv und wird werktags bis 22.00 Uhr realisiert. Problematisch ist auch die Planung des 3. und 6. Seminar bei parallelen Seminargruppen.

Das Problem der UIC-Vergabe wurde gelöst durch Vergabe von Seminargruppen-UIC und Arbeits-UIC für jedes Terminal. Auf Grund der Vielzahl der Dateien mit relativ kleinem Umfang wurde die Praktikumsplatte mit großem Indexfile initialisiert (für 1200).
Die Erfahrung zum "Anwendersystem" kann im Unterricht nicht vermittelt werden (da das System zu weit verzweigt ist), sondern sie wird im abendlichen individuellen Experimentieren gewonnen. Der durch ein Anwendersystem geführte Student baut schnell seine Anfangsscheu ab und ist relativ früh fähig, mit der Anlage zu arbeiten, blockiert jedoch während seiner "Lehrzeit" das Terminal.
Ein mit Systemunterlagen Lernender benötigt eine längere Einarbeitszeit, ist aber in seinen Betriebssystemkenntnissen wendiger.

5. Weitere Vorhaben

Auf der Basis des vorgestellten Programmpaketes DIRAS, das in der ersten Ausbaustufe die Aufgaben des zentralen Steuermoduls (DIALOG) ausführt, wird das Anwendersystem durch Lehrprogramme und Leistungskontrollen ergänzt. Der Anschluß einer Methodenbank (im RZ der TU Dresden in Arbeit) ist vorgesehen. Lehrprogramme sollen insbesondere Probleme behandeln, mit denen der Student bei selbständiger Nutzung ohne Konsultationsmöglichkeit konfrontiert werden kann. Die Verdichtung von zu speichernden Informationen soll den erforderlichen Speicherbedarf von Textdateien, sequentiellen Ursprungsdaten der Lehrprogramme und der Direktzugriffsdateien senken. Außerdem ist beabsichtigt, auf die sequentielle Ursprungsdatei gänzlich zu verzichten, indem ein spezielles Dientsprogramm ausgewählte Bilder der Direktzugriffsdatei zur Korrektur mit EDI bereitstellt.
Dem Anliegen des Schutzes bestimmter Informationen soll Rechnung getragen werden, indem vom Anwendersystemadministrator Teilgraphen und spezielle Knoten gesperrt werden können.
Die Einbeziehung digitalgrafischer Komponenten ist beabsichtigt.

Die Unterstützung der Arbeit des Lehrenden als Autor von Lehrprogrammen oder Leistungskontrollen soll durch das Programmpaket AUTOR verstärkt werden. Eine Auswertekomponente AUSWER soll als Rückkopplung zwischen Lehrenden und Lernende dienen.

Abkürzungsverzeichnis

BYE	- MCR-Kommando zur Abmeldung
CAI	- Computer Aided Instruktion
DIRAS	- Dialograhmen CM 4
EDI	- Editor
FELI	- Fehlerlistenaufbereitung
FLI	- Dienstprogramm zur bildschirmweisen Textausgabe (File-Listing mit Interrupt)
FOR	- FORTRAN IV-Compiler
HEL	- MCR-Kommando zur Anmeldung
HIPO	- Hierarchy Input Process Output
MCR	- Kommandoprogramm (Monitor-Consol-Routine)
PAP	- Programmablaufplan
PIP	- Dateitransferprogramm (Peripheral Interchange Program)
RETTEN	- Indirektkommandodatei zum Speichern (Retten) aus dem Arbeitsbereich in den Speicherbereich der Seminargruppe
RGU	- Rechnergestützter Unterricht
RUN	- Start einer Task
STA[RT]	- Anwendersystem für lernende Nutzer
TKB	- Taskbuilder

Literatur

/1/ Seidel, H.
Anwendersystem für lernende Nutzer.
Unveröffentlichter Bericht. IH Zwickau, ORZ 26.10.1984

/2/ Conrad, E.
Dialograhmenprogramm für K 1600 DIRAK.
Programmdokumentation. IH Zwickau, ORZ, 20.06.1984

/3/ Conrad, E.
Dialogprogrammgeneratoren für Rechenanlagen K 1600 (DPG).
Programmdokumentation. IH Zwickau, ORZ, 20.06.1984

/4/ Kade, G.
Aufbereitung der FORTRAN-Übersetzungsliste.
Beschreibung für den Anwender. IH Zwickau, ORZ 15.10.1984

Friedrich, Steffen

Zu Kriterien für die Beurteilung von Lehrprogrammen

Die gegenwärtige Entwicklung auf dem Gebiet der Mikroelektronik erfaßt wahrhaft alle Bereiche der Volkswirtschaft und des gesellschaftlichen Lebens. Es ist daher mehr als natürlich, daß auch im Bereich der Bildung sich Bemühungen verstärken, um eine möglichst schnelle Nutzung neuester Entwicklungen zu erreichen. Es liegen sicher eine Reihe positiver Effekte für die Ausbildung und die Erziehung unter Einbeziehung dieser neuen Erkenntnisse auf der Hand. Die Integration von EDV und Mikroelektronik im Bildungswesen hat dennoch einen anderen Charakter als einfach die Übernahme einer neuen Technologie. Mit der Verwirklichung dieser Aufgabe werden erste Schritte zur Automatisierung geistiger Arbeit auch in diesem Bereich gegangen. Notwendigerweise ergeben sich daraus auch einige neue Fragestellungen, die gegenwärtig noch nicht klar beantwortet werden können, wie z. B.

- In welcher Weise und wann sollten Computer in der Volksbildung genutzt werden?
- Wie müssen wir uns langfristig darauf vorbereiten?
- Wie ist gesichert, daß damit auch eine größere Effektivität erreicht wird?

Bereits jetzt wird deutlich, daß zur Lösung der anstehenden Probleme eine intensive interdisziplinäre Arbeit notwendig ist, die die Fachgebiete Informatik, Lernpsychologie, Didaktik, Fachmethodik, KI-Forschung u. a. umfaßt.
Diese Arbeit wird insbesondere folgende Richtungen überdecken:

a) Lehren und Lernen mit Computern, d. h., der Computer ist Mittel im pädagogischen Prozeß.

b) Lehren und Lernen von Computern, d. h., der Computer und seine Funktionsweise sind Gegenstand des pädagogischen Prozesses.

c) Testen und Prüfen mittels Computer, d. h., es liegt

ein spezieller Anwendungsfall vor.

d) Verwalten mit Computern, d. h., es erfolgt eine Nutzung für spezielle Hilfsprozesse.

Sicherlich müssen dabei Tendenzen, die diese Entwicklung fördern bzw. hemmen, berücksichtigt werden, um zu erreichen, daß das "Lernen auf Vorrat" durch solche modernen Inhalte und Methoden abgelöst wird, die den Anforderungen an den Werktätigen der nächsten Jahrzehnte entsprechen.

Diese allgemeinen Standpunkte umzusetzen heißt, konkret über Lösungswege eines sinnvollen Computereinsatzes im Bildungsbereich zu diskutieren. Dabei kann eine Konzentration auf folgende 3 Richtungen erfolgen:

(1) COMPUTERVERSTÄNDNIS,
im Sinne der Motivation und der Erzeugung positiver Haltungen zum Einsatz von Computern und damit gleichzeitig der Beseitigung "maschinenstürmerischer Zweifel"

(2) COMPUTERBILDUNG,
im Sinne der Beseitigung des Unvermögens zur Benutzung von informationsverarbeitenden Maschinen und der Herausbildung dafür notwendiger neuer Denkweisen. Bei der Entwicklung von Fähigkeiten und Fertigkeiten zur Benutzung von Computern ergibt sich eine Unterscheidung zwischen dem vorrangigen Einsatz als Mittel bzw. Gegenstand im pädagogischen Prozeß

(3) COMPUTERNUTZUNG,
im Sinne der Anwendung für eine bereits ausgearbeitete Aufgabenstellung, um diese mit dem Mittel Computer schneller und besser zu lösen

Diese aufgezeigten Richtungen dürfen allerdings nicht dazu führen, daß die Einsatzmöglichkeiten von vornherein beschränkt und nicht in der ganzen Breite gesehen werden.

Eine Möglichkeit, Computer als Mittel im Prozeß der Bildung und Erziehung einzusetzen, sehen wir in Lehrsystemen auf der Basis von Mikrorechnern (vgl. /1/, /2/).
Ausgehend von Buchprogrammen hat es in den zurückliegenden Jahren vielfältige Entwicklungen vor allem auf Großrechnern gegeben, die aber in verschiedener Weise auf Grenzen gestoßen sind. Das näher anzuführen, ist nicht unsere Absicht. Es hat sich aber gezeigt, daß auf der Basis von Mikrorechnern neue Schritte möglich werden. Dabei werden neue Fragestellungen bedeutsam, die die Nutzung von Computern mit dieser Zielrichtung betreffen:

a) Wie müssen Computer aussehen, die die Einsatzforderungen als Lehrsystem erfüllen können?
(Das ist die Frage nach der Teachware des Systems, d. h. nach der technischen Gewährleistung und damit eine ingenieur-technische Aufgabe.)

b) Wie müssen Programme gestaltet sein, damit der beabsichtigte Effekt erreicht wird?
(Das ist die Frage nach der Courseware des Systems, d. h. nach der inhaltlichen Gewährleistung und damit eine vor allem pädagogische Aufgabe.)

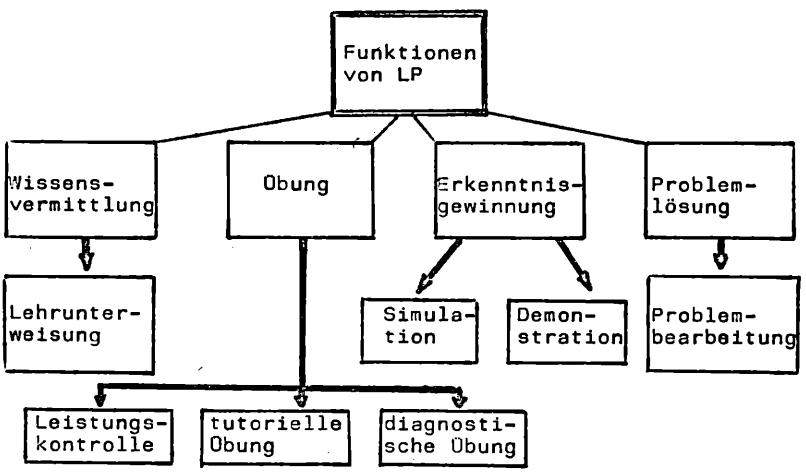
Ohne auf den Teil der Entwicklung der Teachware genauer einzugehen, soll doch die Bedeutsamkeit an folgenden Gesichtspunkten verdeutlicht werden:

- Lehrsysteme sollten auf der Basis von Mikrorechnern entstehen, die Großserien entstammen
- Sicherung einer einfachen Bedienung, vor allem durch ein Minimum an systemsichernden Tätigkeiten
- durch vorhandene Systemfunktionen eine hohe Intelligenz erzeugen
- moderne Softwareentwicklungen zur weiteren Verbesserung nutzen

Solche und ähnliche Gesichtspunkte wurden bei der Entwicklung des Systems MLS/BC weitgehend beachtet (vgl. /3/, /4/).

Immer aktueller und bedeutsamer wird gegenwärtig die Frage nach der Entwicklung von Courseware, nach der Einordnung, Beurteilung und Nutzung entstandener Lehrprogramme.
Mit dem Vorhandensein einer entsprechenden technischen Basis werden natürlicherweise Lehrprogramme entstehen, die nicht alle diesen Status verdienen. Damit wird eine stärkere Differenzierung dringend erforderlich, denn es werden mit Lehrsystemen und zugehörigen Lehrprogrammen Erwartungen erzeugt und so auch Forderungen an das System gestellt.

Eine differenzierte Beurteilung der Courseware kann Oberspitzungen abbauen und einen erfolgreichen Einsatz bewirken.
Wir fassen Lehrprogramme als das Ergebnis der vom Pädagogen vorgeplanten Lehrhandlung auf, das durch adäquate Mittel dargestellt ist und den Rahmen für die Kommunikation des Benutzers mit dem Lehrsystem vorgibt.
Funktionen, die Lehrprogramme unter didaktischer Sicht erfüllen können, stellen sich u. E. wie folgt dar:



Zur genaueren Charakterisierung sollen Antworten auf 3 Fragen dienen:

(1) Wie ist der Aufbau des Gesamtprogramms?
(2) Was geschieht durch Lernenden?
(3) Wie ist die Lernersteuerung?

Frage	Lehrunterweisung	Leistungskontrolle	tutorielle Übung	diagnostische Übung	Simulation	Demonstration	Problembearbeitung
(1)	standardisierte Programmstruktur mit Haupt- und Nebenpfaden	standardisierte Programmstruktur mit Punktbewertung	variable Struktur mit Hauptlehrpfad	variable Struktur an Leistungen angepaßt	beliebiges Einpassen in vorhandene Programmstrukturen	feste Programmstruktur mit sehr variablen Datensatz	keine Programmstruktur
(2)	Abarbeitung programmierter Instruktionen	Beantwortung von Fragen, Lösung von Aufgaben	Festigung von Kenntnissen durch geführte Aufgabenlösung	Beseitigung von Fehlerursachen durch Aufgabenlösung	selbst. Experimente an Modellen oder Modelldarstellung	Beobachtung von Computerexperimenten	Suche von Lösungen mittels Computer
(3)	feste Steuerung durch Lehrsystem	feste Aufgabenfolge	resultatsabhängige Steuerung durch Lehrsystem	diagnoseabhängige Steuerung durch Lehrsystem	lerner- od. systemgesteuerte Unterprogramme	keine Lernersteuerung	Steuerung durch die Auswahl von Modulen

Mit der Übersicht wird ein Ansatz für Kriterien zur Einordnung von Lehrprogrammen nach ihrer Funktion vorgestellt. Durch zusätzliche Fragestellungen ist diese noch zu erweitern. Das erschwert allerdings eine praktikable Klassifizierung, die künftigen Lehrprogrammanwendern helfen soll.
Wir vertreten die Auffassung, daß eine mitunter angestrebte rein quantitative Charakterisierung von Lehrprogrammen zwar eine Einordnung liefert, aber einen effektiven Zugriff erschwert. Es erscheint sinnvoll, durch Nutzung von Dateiprojekten einen mehr qualitativen Zugriff auf Lehrprogramme zu organisieren. Aus den Nutzungsraten sind direkt Konsequenzen für die weitere Programmentwicklung ableitbar.
Das betrifft vor allem:

- Lehrinhalte und mögliche Lehrstrategien
- Grundmodelle für LP-Typen
- Systematisierung von Lehrinhalten
- Überlegungen zum Lerndialog, insbesondere einer brauchbaren Lernersteuerung

Damit ergeben sich auch weitere Arbeitsrichtungen, die im Ergebnis dazu beitragen sollen, entstehende Lehrprogramme effektiv einzuschätzen, zu ordnen und den Zugriff darauf zu organisieren.

Literatur

/1/ Gahmig/Kerner
MLS/BC - ein mikrorechnergestütztes Lehrsystem zum Einsatz in der Volksbildung
In: INFO 84, Tagungsberichte, Teil 2, Dresden 1984

/2/ Kerner/Gahmig/Friedrich
Erarbeitung und Nutzung des Lehrsystems MLS/BC
In: 29. IWK der TH Ilmenau 1984

/3/ Berichte zum 2. Seminar "Computer als Mittel und Gegenstand der Ausbildung", Ottendorf-Okrilla 1984
In: Dresdner Reihe zur Forschung 13/84

/4/ MLS/BC - ein mikrorechnergestütztes Lehrsystem auf der Basis von Bürocomputern.-
In: Dresdner Reihe zur Forschung 6/85

Autor:

Friedrich, Steffen
Dr. paed.
Pädagogische Hochschule Dresden
Sektion Mathematik, WB Informatik
8060 D r e s d e n
PSF 365

Gahmig, Frieder

Erfahrungen bei der Implementierung und Nutzung des mikrorechnergestützten Lehrsystems MLS/BC

Seit mehr als zwei Jahren werden durch die Forschungsgruppe "Computer in pädagogischen Prozessen" der Pädagogischen Hochschule "Karl Friedrich Wilhelm Wander" Dresden u. a. Probleme der Computeranwendung zur individuellen Steuerung eines pädagogischen Prozesses bearbeitet. Dabei stand die Aufgabe, basierend auf entsprechenden Voruntersuchungen, nicht nur Lösungen des computergestützten Unterrichts der 70er Jahre erneut zu einem System zusammenzuführen, sondern ein System zu entwickeln, das für die Einbindung neuer Mittel und Methoden weitgehend offen ist.
Besonderer Wert wurde auf die Planung und Realisierung der Schnittstelle Mensch-Maschine gelegt. Diese Schnittstelle besteht in dreierlei Hinsicht (Lehrprogrammautor-Maschine, Lernender-Maschine, Bediener-Maschine).
Darüber hinaus wurden bei der Konzeption von MLS pädagogisch und didaktisch begründete Modelle und Standpunkte zu Fragen der lernverlaufsabhängigen Lenkung der Antwortkontrolle und der pädagogischen Diagnose einbezogen.
Die Konzeption wurde weiterhin beeinflußt durch moderne Verfahren und Methoden der künstlichen Intelligenz, die in Lehrsystemen effektvoll einsetzbar sind (Formelmanipulation, Computersimulation, Computerdiagnostik).
Schließlich beeinflußten eine Reihe progmatischer Aspekte bzw. Erfahrungen anderer Lehrsystemlösungen die Art und Weise der Realisierung von MLS.

1. Die Anwendungsbreite und Anwendungshäufigkeit hängt wesentlich von der Verfügbarkeit der Basismaschinen für Lehrsysteme ab.
   Deshalb wurde MLS in der Variante MLS/BC für Bürocomputer implementiert, die serienmäßig hergestellt werden.

2. Der Anwendungseffekt und die Nachnutzbarkeit des Systems hängt ab von Qualität und Aktualität der das System stützenden Betriebssoftware, von der Qualität der Lehrprogramme

und von der Qualität der speziell für das Lehrsystem existierenden Einfahrhilfen.

Die wohl wichtigste Fragestellung für Lehrsysteme ist die, wie der das Lehrsystem einsetzende Lehrer die entsprechenden Programme schreiben kann. Zur Unterstützung dieses Prozesses wurde eine spezielle Lehrprogrammformulierungssprache (LEFO) entwickelt.
Lehrprogramme für Computer, selbst eingebettet und eingesetzt in einen globalen didaktischen Prozeß, zielen auf einen durch Einsatzgebiet, Lehrziele und Lehrmethoden charakterisierbaren lokalen Lehr- und Lernprozeß.
Dafür müssen sowohl Bedingungen seitens des gewährleistenden Systems als auch seitens des Lehrprogramms erfüllt sein.
Dieser speziellen Zielsetzung Rechnung tragend, wurde die Sprache "LEFO" entwickelt, die auf einem pädagogisch begründbaren Lenkungsmodell einer höheren Anforderungen genügenden Antwortkontrolle und einem speziellen Datenkonzept basiert.
Charakteristisch für LEFO ist:

- die strenge Trennung zwischen dem Zeitpunkt und dem Ort der Eingabe und der Verwendung der Lehrprogrammtexte,
- der parametrisierte Sprachaufbau,
- die gute Lesbarkeit der LEFO-Befehle,
- das fachsprachenähnliche Übersetzungsprinzip mittels Vorübersetzern, das eine weitgehende Portabilität von LEFO gestattet,
- die Umsetzung der LEFO-Befehle durch Semantikbausteine (Prozeduren), die in höheren Programmiersprachen (z. B. PASCAL) geschrieben werden.

Die LEFO-Grundbefehle stellen komplexe Bausteine für nahezu alle dankbaren pädagogisch-methodischen Handlungen dar.
LEFO ist eine pädagogisch-methodische Fachsprache.

Für MLS wurden eine Reihe von Modellen entwickelt, die durch ihre Umsetzung in MLS/BC bereits eine gewisse Fundierung erfahren haben. Diese Modelle sind offen und für andere Lösungen übertragbar. Sie bilden die Grundlage, wenn im mündlichen Vortrag mehr über ein <u>portables mikrorechnergestütztes Lehrsystem</u> gesagt werden soll.

Mit MLS/BC wird für das Volksbildungswesen der DDR eine Vorlaufentwicklung betrieben.
Es ist nicht zu erwarten, daß Bürocomputer für diesen Zweck in größeren Stückzahlen in den Oberschulen der DDR zum Einsatz kommen werden. Für die dagegen verfügbaren Heimcomputer ist das hier vorgestellte MLS noch nicht einsetzbar. Diesen Homecomputern fehlen noch Diskettenlaufwerke und Drucker als Hardwareergänzung sowie ein leistungsstärkeres Betriebssystem mit einem PASCAL-Compiler als Softwareergänzung. Es darf aber angenommen werden, daß diese Ergänzungen entsprechend zum internationalen Trend verfügbar sein werden. Dann hat man auch bald eine MLS/HC (Heimcomputer)-Variante!

Gahmig, Frieder; Dr. rer. nat.
Pädagogische Hochschule
"Karl Friedrich Wilhelm Wander" Dresden
Sektion Mathematik
WB Rechenstelle
8060 D r e s d e n
Wigardstraße 17, PSF 365

Dr. sc. Gernert, Hans

Rechnergestütztes Planspiel KOMBINAT in der Aus- und Weiterbildung von Führungskadern

1. Rechnergestützte Planspiele zur Nachahmung der Verhaltensweise realer ökonomischer Prozesse

Rechnergestützte Planspiele sind als eine interdisziplinäre Form der Untersuchung und Darstellung komplexer ökonomischer Vorgänge anzusehen. Der handelnde Mensch steht bei ihnen im Mittelpunkt des Interesses. Die Teilnehmer können in die Rolle des Lehrenden, des Lernenden oder des Forschenden schlüpfen und im Spiel bei hinreichender Komplexität des Gegenstandsbereiches ganz unterschiedliche Formen der Zusammenarbeit entwickeln. So lassen sich Entscheidungssituationen unter mehr oder weniger großer Ungewißheit simulieren, die Teilnehmer können Bedingungen des sozialistischen Wettbewerbs vorfinden oder sich im Rollenspiel innerhalb von Leitungshierarchien bewegen.

In einem Planspiel stellen die Spielteilnehmer mit ihren Entscheidungen einen aktiven Bestandteil dar, den sog. Aktionsbereich.
Der Gegenstand der ökonomischen Analyse im Spiel, d.h. die konkrete Situation, auf die die zu treffenden Entscheidungen angewandt werden, wird durch das Simulationsmodell simuliert. Auf Grund der Entscheidungen wird das Modell von einer Ausgangssituation in einen neuen Zustand überführt. Das Simulationsmodell wird daher im Planspiel als Reaktionsbereich bezeichnet.

Die Dynamik realer Prozesse wird im Planspiel nunmehr durch eine Aufeinanderfolge von Entscheidungsperioden, also eine zeitliche Sequenz von Wechselwirkungen zwischen Aktions- und Reaktionsbereich nachgeahmt.

Die Entwicklung komplexer Planspiele steht in engem dialektischem Zusammenhang zur Entwicklung der ökonomischen Theorie.

Zur Simulation ökonomischer Prozesse in einem Planspiel sind erforderlich:

1. Die qualitative Beschreibung der ökonomischen Erscheinungen. So hängt z.B. die Entwicklung der Arbeitsproduktivität in einem Betrieb u.a. ab vom Einsatz von Wissenschaft und Technik, der Qualifikation der Arbeitskräfte, der Organisation des Produktionsprozesses usw..
2. Die quantitative Bestimmung dieser ökonomischen Gesetzmäßigkeiten.
   Es muß also z.B. ermittelt werden, wieviel % Steigerung der Arbeitsproduktivität durch welchen Aufwand an finanziellen oder materiellen Mitteln erreicht wird.
3. Die durchgängige Darstellung aller für den realen Prozeß und die Einsatzziele des Spieles relevanten Zusammenhänge in Form von Algorithmen, die nach entsprechender Programmierung auf einer Rechenanlage abgearbeitet werden müssen.

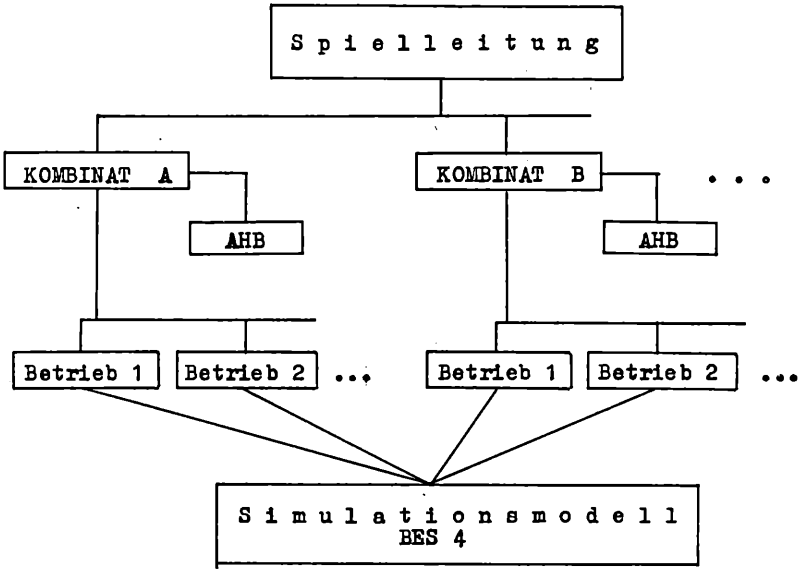
Hieraus ergibt sich die Forderung nach einem hohen Entwicklungsniveau der vom Gegenstandsbereich des Spiels berührten Wissenschaftsdisziplinen (Betriebswirtschaft, Volkswirtschaft, Finanzen usw.) und zugleich eine enge Verbindung mit Methoden und Verfahren der Analyse ökonomischer Prozesse (Mathematik, Statistik).
Für die technische Realisierung komplexer Spiele wurden bisher überwiegend Rechenanlagen im off-line Betrieb mit Datenerfassung über Lochkarte oder Lochband sowie Ausgabe der Ergebnisse über Druckliste eingesetzt.

Die Verfügbarkeit von Bildschirmtechnik bei Fernverarbeitung mit Großrechnern oder im Anschluß an leistungsfähige Kleinrechentechnik schafft gegenwärtig neue Möglichkeiten der Nutzung von Planspielen. Das bedingt aber zugleich auch die Entwicklung neuer Konzeptionen zur Gestaltung von effektiven Einsatzformen solcher Spiele für die Aus- und Weiterbildung von Wirtschaftskadern.

2. Mehrebenenplanspiel KOMBINAT

2.1. Übersicht über die Struktur des Spiels

Im Planspiel KOMBINAT nehmen die Teilnehmer Verantwortlichkeiten in mehreren hierarchisch zueinander stehenden Leitungebenen wahr.



Die Anzahl der Kombinate im Spiel sowie die Zahl der Betriebe in jedem Kombinat ist frei wählbar. Vor Spielbeginn sind jedoch von der Spielleitung entsprechende Datenbasen zu generieren.

Die Betriebe können bei Spielbeginn von der gleichen Ausgangssituation ausgehen. Es ist jedoch auch möglich und aus Gründen der Ausarbeitung differenzierter Kombinatsstrategien zu empfehlen, solche Ausgangssituationen zu generieren, bei denen sich die Betriebe innerhalb der Kombinate unterscheiden.

Das Planspiel kann bis zu 6 Perioden umfassen, wobei in jeder

Periode ein Zeitraum von 6 Monaten simuliert wird.
Es ist jedoch auch möglich, Perioden zu wiederholen oder von einer bestimmten Situation als Ergebnis eines Simulationslaufes aus mehrere Entscheidungsvarianten für eine Periode durchzurechnen.

Die Spielleitung repräsentiert die zentralen staatlichen Planungs- und Leitungsorgane. Sie ist als das zuständige Industrieministerium anzusehen, vertritt aber zugleich auch andere Verantwortungsbereiche, wie z.B. Organe, die für die Bilanzierung und Realisierung von Außenhandelsgeschäften verantwortlich sind.
Von der Spielleitung erhalten die Kombinate alle Planzahlen und weitere Informationen. Ihr gegenüber rechnen die Kombinate ihre Pläne ab und legen Rechenschaft über die Effektivität der Arbeit in den den Kombinaten nachgeordneten Betrieben.

Auf der Leitungsebene der Kombinate übernehmen Spielteilnehmer die Verantwortlichkeit der Leitung von Industriekombinaten, denen mehrere Betriebe unterstellt sind. Eine spezielle Aufgabe entsteht den Kombinaten in der Wahrnehmung der Funktionen eines ihnen zugeordneten Außenhandelsbetriebes (AHB).

Die niedrigste Leitungsebene des Spiels ist mit der Leitung der Kombinatsbetriebe beauftragt. Die hierbei getroffenen Entscheidungen sind Eingangsdaten für Simulationsläufe, bei denen das Simulationsmodell BES 4 (GERNERT/MESSERSCHMIDT, HUB, 1984) das Verhalten der betreffenden Betriebe simuliert. Dieses Modell ist eine Weiterentwicklung des Modells BES 1 (GERNERT/KÖLZOW, HUB, 1974).

Das Planspiel KOMBINAT ist modular aufgebaut. Es kann in der hier beschriebenen Form durchgeführt werden. Es ist jedoch auch möglich, mit Hilfe des Modells BES 4 nur die betriebliche Leitungsebene zu simulieren. Aus dieser An-

wendungsvariante ergeben sich dann Planspielformen, die dem Planspiel BES 1 formal entsprechen.

Gleichzeitig kann das Modell KOMBINAT unter Angliederung von zwei Teilmodellen, die ausgewählte Bereiche der Wirtschaft Großbritanniens beschreiben, zu einem internationalen Handelsspiel DDR-Großbritannien ausgebaut werden.

2.2. Leitungsebene KOMBINAT

In ihrer Funktion als wirtschaftsleitendes Organ übergibt die Spielleitung der Kombinatsleitung folgende Unterlagen:

- Bedarfszahlen für abzusetzende Erzeugnisse in der Folgeperiode, untergliedert nach Produkten und Absatzgebieten. Die Exporte in das NSW sind dabei Ergebnis der Verhandlungen des AHB (Menge, Termine, Preise).
- Planzahlen für jeweils ein Planjahr, aufgegliedert nach Perioden.
- Volkswirtschaftlich bilanzierte Ressourcen auf dem Gebiet der Investitionstätigkeit.
- Angaben zur Finanzierung von Investitionen und von Maßnahmen auf dem Gebiet von Wissenschaft und Technik.
- Weitere Informationen, die sich aus aktuellen volkswirtschaftlichen Entwicklungen ergeben.

Für die die Kombinatsleitung repräsentierenden Spielteilnehmer ergeben sich zwei Aufgabenbereiche:

1. Entscheidungen zur Planung und Leitung der dem Kombinat nachgeordneten Betriebe.
   Jeder dieser Betriebe ist im Rahmen des Kombinatsverbandes selbständig. Die Planauflagen des Industrieministeriums (Spielleitung) sind daher auf diese Betriebe aufzuschlüsseln. Dabei ist die jeweilige wirtschaftliche und technologische Situation dieser Betriebe zu beachten.
   Das Kombinat bildet und verwendet zentralisierte Fonds, um seine mit der volkswirtschaftlichen Aufgabenstellung übereinstimmende Wirtschaftsstrate-

gie langfristig durchsetzen zu können.
Der Investfonds wird aus Amortisationsanteilen der Betriebe gebildet. Diese Mittel können in Übereinstimmung mit der Investitionsstrategie des Kombinates an die nachgeordneten Betriebe umverteilt werden.
Gleiches gilt für den Fonds Wissenschaft und Technik.
Die Spielleitung kann in ihrer Rolle als Ministerium beiden Fonds weitere Mittel zu führen bzw. Mittel abziehen.

2. Realisierung der Arbeit des Außenhandelsbetriebes.
Der zweite Entscheidungsbereich der Kombinatsleitung umfaßt alle Import- und Exportaktivitäten des gesamten Kombinates.
Dazu hat der AHB folgende Aufgaben durchzuführen:
   - Ermittlung des voraussichtlichen Bedarfes für die produzierten Erzeugnisse im RGW-Bereich und in Großbritannien.
   - Abschluß von Lieferverträgen nach Menge, Preis, Qualität und Zielland je Erzeugnis für jede Simulationsperiode.
   - Verhandlung der Importbedingungen für Roboter nach Menge, Preis (VGW) und Importabgabepreis bei Einhaltung des bestätigten Importvolumens insgesamt.
   - Ermittlung der Ergebnisse der Exporttätigkeit des AHB für jede Periode.

Der AHB des Kombinates arbeitet nach dem Prinzip der Wirtschaftlichen Rechnungsführung. Für jede Simulationsperiode sind daher ausgewählte Kosten und Erlöse aus der außenwirtschaftlichen Tätigkeit des AHB zu berechnen und zusammenzustellen.

Die Kombinatsleitung ist nach jeder Periode gegenüber dem Ministerium rechenschaftspflichtig und nimmt die Rechenschaftslegung der nachgeordneten Betriebe entgegen.

2.3. Leitungsebene Betrieb

Für jeden Kombinatsbetrieb sind in jeder Periode Entscheidungen auf den Gebieten Produktion, Wissenschaft und Technik, Ausrüstungen, Arbeitskräfte, Material sowie Bildung und Verwendung betrieblicher Fonds zu treffen. Diese Entscheidungen sind aus der Ausgangssituation bzw. der Vorperiode und aus den Planauflagen der Kombinatsleitung abzuleiten.

Die folgenden Ausführungen enthalten einige Informationen über den mit dem Modell BES 4 simulierten betrieblichen Prozeß. Sie sind den Komplexen Produktionssortiment, Ausrüstungen, Arbeitskräfte, Material und Fondsbildung zugeordnet.

Produktionssortiment:

Zu Beginn des Planspiels produziert jeder Kombinatsbetrieb zwei Erzeugnisarten, die sich als PKW (E1) und komplette Sätze von Ersatzteilen dazu (E2) interpretieren lassen. Während des Spiels wird ein neues PKW-Modell (E3) mit den dazugehörigen Ersatzteilen (E4) in die Produktion eingeführt.

Das Kombinat verkauft seine Erzeugnisse auf drei Absatzgebieten (DDR, RGW, Großbritannien).

Jedes Erzeugnis kann in drei Güteklassen produziert werden.

Die Aufgabe der Entscheidungstätigkeit besteht in der Gewährleistung einer bedarfsgerechten Produktion bei Einhaltung aller Plankennziffern.

Ausrüstungen:

Um die Planauflagen der Kombinatsleitung zur Steigerung der Effektivität der Produktion, zur Erreichung des Exportergebnisses und einer bedarfsgerechten Produktion realisieren zu können, ist das technologische Niveau der Produktion wesentlich zu erhöhen.

Durch gezielte Investitionsmaßnahmen ist es möglich, eine bestimmte Anzahl von produzierenden Einheiten durch rekonstruierte Maschinen der DDR-Produktion und durch Roboter aus NSW-Importen zu ersetzen.

Die Struktur und der Umfang der vorzunehmenden Investitionen ist durch die Planaufgaben zu begründen und hängt von der Zustimmung der für die Bilanzierung des Betriebes zuständigen Kombinatsleitung ab.

Arbeitskräfte:

Für eine Erarbeitung der Kapazitätsbilanz ist eine Gegenüberstellung des Arbeitsvermögens der Arbeitskräfte und der Inanspruchnahme dieses Zeitfonds durch das geplante Produktionsvolumen erforderlich.

Die zur Herstellung einer Einheit eines Erzeugnisses erforderlichen Arbeitsstunden sind zu Spielbeginn der Ausgangssituation zu entnehmen. Diese Angaben verändern sich während des Spielverlaufes unter Einwirkung der jeweils realisierten Maßnahmen im Bereich Wissenschaft und Technik. Hierbei werden wirksam: Forschungsmittel, Ausgaben für WAO, Qualifizierung der Arbeitskräfte, Verschleißquote, Einsatz rekonstruierter Ausrüstungen und insbesondere von Robotern, Ausschußquote.

Durch gezielte Freisetzung von Arbeitskräften, Neueinstellungen und Maßnahmen zur Verbesserung der Arbeits- und Lebensbedingungen kann das Arbeitsvermögen des Betriebes mit den Planaufgaben in Übereinstimmung gebracht werden.

Material:

Die zur Herstellung einer Einheit eines Erzeugnisses erforderlichen Mengeneinheiten sind der Ausgangssituation zu entnehmen.

Während des Spiels läßt sich dieser Aufwand durch Maßnahmen der Intensivierung des Produktionsprozesses verändern. Hierbei werden wirksam: Forschungsmittel, Verschleißquote, Einsatz von rekonstruierten Ausrüstungen und Robotern, Ausschußquote.

Die Entscheidungen für Materialbestellungen sind mit den Produktionsaufgaben der entsprechenden Perioden abzustimmen, damit der Betrieb mit ökonomisch vertretbaren Materialbeständen arbeiten kann.

Fondsbildung:

Zur umfassenden Intensivierung des Reproduktionsprozesses sozialistischer Industriebetriebe sind Bildung und Verwendung des Fonds Wissenschaft und Technik sowie des Investitionsfonds von besonderer Wichtigkeit.

Beide Fonds werden in den durch BES 4 simulierten Betrieben gebildet. Die Kombinatsleitung hat die Möglichkeit, Mittel aus diesen Fonds zwischen den ihr unterstellten Betrieben so umzuverteilen, daß ein effektiver Einsatz aller Ressourcen möglich wird.

3. Einsatzmöglichkeiten und -bedingungen

Komplexe Mehrebenenplanspiele bieten eine gute Möglichkeit zur Intensivierung der Aus- und Weiterbildung von Wirtschaftskadern, weil man mit ihrer Hilfe im Rahmen des Ausbildungsprozesses und ohne Übernahme des Risikos materieller Verluste ausgewählte Leitungsaufgaben demonstrieren und ihre Lösung üben kann.

Zur Durchführung des Planspiels KOMBINAT werden mehrere Teilnehmergruppen gebildet, für deren Zusammensetzung folgende Empfehlungen gelten:

- jedes Betriebsleitungskollektiv 3 - 4 Teilnehmer,
- Kombinatsleitung ca. 4 Teilnehmer,
- In Abhängigkeit von der Zahl in einem Spiel geführter Kombinate und Betriebe sollte sich die Spielleitung aus 2 - 3 Personen zusammensetzen.

Als Mehrebenenplanspiel orientiert sich KOMBINAT stark auf die Diskussion und die Verhandlungsführung zwischen den Teilnehmern, die sich aus umfassenden Analysen der jeweiligen Simulationsergebnissen ableiten.
Damit ergibt sich eine sehr aktive und infolge ihrer Dynamik und Komplexität auch praxisnahe Lehrform. Es werden keine Einzelkenntnisse 'abgefragt', sondern die Teilnehmer müssen es lernen, ökonomische Prozesse in ihren Wechselwirkungen zu analysieren und planmäßig zu leiten.

Das von K. Messerschmidt (ORZ der HUB) für das Modell BES 4 erarbeitete umfangreiche Programmpaket enthält ca. 4500 FORTRAN-Anweisungen. Die für die Abarbeitung des Programms erforderliche Hauptspeicherkapazität beträgt ca. 76 KByte.

Für die Eingabe der Entscheidungen bzw. die Ausgabe der Ergebnisse werden Bildschirmgeräte des AP 64 unter Nutzung des TSO eingesetzt, die an ESER-Rechner EC 1022 bzw. EC 1055 angeschlossen sind.

Das Planspiel KOMBINAT wurde während des 10. Internationalen Seminars über rechnergestützte Planspiele vorgestellt, das von der Sektion Wirtschaftswissenschaften der Humboldt-Universität zu Berlin im Dezember 1984 durchgeführt wurde. Es steht zur Nachnutzung zur Verfügung.

Verfasser:
Doz. Dr. sc. oec. Hans Gernert
Humboldt-Universität zu Berlin
Sektion Wirtschaftswissenschaften
Bereich Statistik

1020 Berlin
Spandauer Str. 1

Hartwig, W.-H.

Grundpositionen und erste Erfahrungen zum Einsatz des Lehrsystems AOS VUZ im Lehr- und Studienprozeß an der TU Dresden

Die zunehmende Verbreitung moderner Informationstechnologien zur Automatisierung geistiger Prozesse als einer Schlüsselposition des wissenschaftlich-technischen Fortschritts in der gesamten Volkswirtschaft führt zu bedeutsamen Konsequenzen für Inhalt und Gestaltung von Ausbildungsprozessen an den Hoch- und Fachschulen unseres Landes. So wird es vor allem erforderlich, praktisch alle Studenten im Verlaufe ihres Studiums zur Beherrschung derjenigen rechnergestützten Arbeitsweisen zu führen, die für die Tätigkeit in ihrer späteren beruflichen Praxis charakteristisch sind bzw. zukünftig sein werden. Zum anderen gilt es, die Möglichkeiten der modernen Rechentechnik zielstrebig zu nutzen, um ausgewählte Lehr- und Aneignungstätigkeiten von Hochschullehrern und Studenten zu intensivieren und so die Effektivität der Ausbildung im ganzen erhöhen zu helfen. Diesem Einsatzfeld von EDVA, das mit Stichworten wie "rechnergestützte Lehrsysteme und -programme" und/oder "computerunterstützter Unterricht" umrissen werden kann, wendet sich der vorliegende Beitrag zu. Er soll dazu dienen, ein hochschulpädagogisches Untersuchungsprojekt vorzustellen, das vom Forschungszentrum für technische Lehr- und Lernmittel der TU Dresden in Zusammenarbeit mit einigen Lehrkollektiven dieser Universität in Angriff genommen wurde.

In den letzten Jahren wurden im Rahmen eines "Komplexprogramms von Arbeiten zur Schaffung rechnergestützter Lehrsysteme auf der Basis von EDVA" des MHF der UdSSR sowie entsprechender Festlegungen im Zielprogramm des Staatlichen Komitees für Wissenschaft und Technik in zahlreichen sowjetischen Hochschulen und Weiterbildungszentren rechnergestützte Lehrsysteme implementiert und zumindest experimentell in den Ausbildungsprozeß eingeführt. Der größte Teil von ihnen basiert auf einer unter Leitung des Moskauer Forschungsinstituts für Probleme des Hochschulwesens (NIIVŠ) entwickelten Familie von Anwenderprogrammpaketen, deren gegenwärtig aktuellste Generation die Bezeichnung AOS VUZ (Abkürzung für die russische Entsprechung von "rechnergestütztes Lehrsystem für Hochschulen") trägt. Sie wurde vom "Wissenschaftlichen Rat für Probleme der Schaffung und Anwendung rechnergestützter Lehrsysteme" beim MHF der UdSSR als Typenprojekt bestätigt und zur breiten Nutzung in der Ausbildungspraxis empfohlen.

Als Voraussetzung dafür wurde u. a. eine umfassende Dokumentation herausgegeben, die Systemunterlagenwartung zentral organisiert und mit der periodischen Herausgabe eines Lehrprogrammkatalogs für AOS VUZ begonnen. Weiterhin sind Anstrengungen zur Qualifizierung von Hochschullehrern als Lehrprogrammautoren eingeleitet und methodische Arbeiten zur Lehrprogrammgestaltung geführt worden. Derzeit ist AOS VUZ an über 100 Bildungseinrichtungen der UdSSR eingeführt, die Zahl der Bestellungen ist noch weitaus höher.

Das Lehrsystem AOS VUZ sichert die Realisierung von Lehrprogrammen im Dialog zwischen Lernendem und System über standardmäßige Bildschirmdisplays verschiedener Typen und unterstützt die Arbeit von Lehrprogrammautoren, aufsichtsführenden Hochschullehrern und Angehörigen des technischen Betreuungspersonals. Für jeden Nutzertyp existieren spezielle Kommunikationsmittel (Kommandosprachen), die auf die von ihm zu lösenden Aufgaben zugeschnitten sind. Lehrprogramme werden mit Hilfe der Autorensprache JAOK auf Quelltextniveau formuliert, eingegeben und getestet. Es handelt sich dabei um eine spezielle Programmiersprache mittleren Niveaus, die sowohl assemblerhafte Züge als auch Merkmale einer problemorientierten Sprache trägt. Durch Möglichkeiten der Verwendung von Makros und Unterprogrammen sowie zusätzlicher, in Maschinensprache geschriebener Funktionen unterstützt sie die Modularität der Programmierung und sichert die Offenheit des Systems gegenüber beliebigen Weiterentwicklungen. AOS VUZ ist in mehreren Versionen für unterschiedliche gerätetechnische Konfigurationen verfügbar, die in der Perspektive das gesamte Spektrum moderner rechentechnischer Mittel umfassen werden, darunter auch Klein- und Mikrorechner. Ausführlichere Informationen zum Lehrsystem AOS VUZ können u. a. /1/ und /2/ entnommen werden.

Im Hochschulwesen der DDR wurden ab Anfang der siebziger Jahre unter koordinierender Beratung des Forschungszentrums für technische Lehr- und Lernmittel der TU Dresden einige Entwicklungsarbeiten auf dem Gebiet rechnergestützter Lehrsysteme betrieben. In ihrem Ergebnis entstanden solche Projekte wie LEDA, MERUS, REGEL und RGU (vgl. z. B. /3/). Ihre pädagogische Erprobung, so begrenzt sie z. T. auch wegen der beschränkten gerätetechnischen Basis bleiben mußte, bewies zumindest zweierlei: Einmal die potentielle Möglichkeit, mit Hilfe solcher Systeme Intensivierungseffekte im Ausbildungsprozeß zu erreichen, zum anderen aber auch die Notwendigkeit, dafür eine Reihe spezifischer Voraussetzungen zu schaffen. Eine davon besteht darin, daß sich der hohe für die Entwicklung

niveauvoller Lehrprogramme benötigte Aufwand nur dann in spürbare pädagogische Effekte umsetzen kann, wenn die breite Nutzung der Ergebnisse gewährleistet ist. Daraus resultieren Forderungen nach Verwendung standardmäßiger Gerätetechnik und Betriebssoftware sowie nach zentralisierter Lehrprogrammentwicklung, die von den oben genannten Lehrsystemen nicht erfüllt werden konnten.

In Auswertung der gesammelten Erfahrungen und unter Berücksichtigung des internationalen Standes wurde zu Beginn der laufenden Fünfjahrplanperiode im Forschungszentrum für technische Lehr- und Lernmittel die Entscheidung getroffen, in enger Forschungskooperation mit dem Moskauer NIIVŠ weitere Untersuchungen zur Entwicklung und zum Einsatz rechnergestützter Lehrsysteme durchzuführen. In Anbetracht des ausgereiften Entwicklungsstandes der Lehrsystemfamilie SPOK-VUZ (Vorgänger von AOS VUZ für DOS/ES, vgl. /4/) und der Notwendigkeit, bei der Fülle der zu lösenden Aufgaben die in den sozialistischen Ländern verfügbaren Kräfte in optimaler Arbeitsteilung einzusetzen, wurde auf weitere Eigenentwicklungen verzichtet und statt dessen die Übernahme von AOS VUZ an die TU Dresden vorbereitet. Als echtes Produkt internationaler sozialistischer Forschungskooperation wurde auf der Grundlage einer in der TU Dresden erarbeiteten "Pädagogisch-technischen Aufgabenstellung zur Einführung des Lehrsystems AOS VUZ in den Experimentalbetrieb an der TU Dresden" vom Moskauer NIIVŠ in Zusammenarbeit mit dem MIIGA (Moskauer Institut für Zivilluftfahrt) die Systemversion AOS VUZ/AP-64 entwickelt, die für den Einsatz an der TU Dresden geeignet ist. Ihre Übergabe durch den sowjetischen Partner erfolgte im Dezember 1984 am Rechenzentrum der TU Dresden.

Als gerätetechnische Basis dient eine EDVA ES-1022, die über 512 KByte Hauptspeicherkapazität verfügt. Über einen Multiplexor MPD 4 vom Typ ES 8404 ist ein Abonnentenpunkt AP-64 mit 5 Bildschirmen und einem Mosaikdrucker angeschlossen, der die Kommunikation der Nutzer mit dem System sichert. An weiteren Ressourcen sind eine Magnetplatteneinheit vom Typ ES-5061 sowie ein bis zwei Magnetbandgeräte erforderlich. Paralleldrucker und Lochkartenleser werden nur zeitweise benötigt. Als Betriebssystem wird die OS/ES-Version 6.1 M4B genutzt.

Dem experimentellen Einsatz der Lehrsystemversion AOS VUZ/AP-64 an der TU Dresden werden eine Reihe von Positionen zugrundegelegt, die sich aus den bisherigen Erfahrungen

des Einsatzes rechnergestützter Lehrsysteme sowie speziell entwickelter Lehrgeräte ergeben und sich z. T. bereits in der Ausbildungspraxis bewährt haben. Einige der wichtigsten sollen hier stichpunktartig angegeben werden.

- Rechnergestützte Lehrsysteme sind ihrer Natur nach technische Gebilde (Einheit von Programm- und Gerätetechnik), ihre Zwecksetzung ist jedoch didaktisch und wird vom jeweiligen Lehrprogramm bestimmt. Pädagogisch begründete Anforderungen an die Gestaltung derartiger Systeme müssen demzufolge von den Forderungen ausgehen, die an durch sie zu realisierende Lehrprogramme zu stellen sind. Dabei werden nicht nur einzelne, sondern bestimmte Klassen von Lehrprogrammen zu berücksichtigen sein.
- Zu den vorrangigen didaktischen Einsatzorten rechnergestützter Lehrsysteme zählen vor allem solche Prozesse im Selbststudium sowie in den Lehrveranstaltungsformen Übung, Seminar und Praktikum, die auf das Festigen von Kenntnissen, das Ausprägen spezieller Fähigkeiten und die Entwicklung geistiger und manueller Fertigkeiten gerichtet sind und dabei besonders der differenzierten Führung der Lernenden bedürfen. In diesem noch relativ weit gefaßten Bereich ist der Lehrsystemeinsatz gegenüber der Verwendung anderer Hilfsmittel vor allem dann als dringlich anzusehen, wenn
  - die Lehrprogrammrealisation mit umfangreichen und/oder komplizierten Aufgaben der Informationsverarbeitung verbunden ist (Generierung von Aufgaben und Texten, Analyse relativ frei formulierter Antworten, Diagnose- und Protokollführungsaufgaben u. a.) und/oder
  - der Rechner gleichzeitig als Gegenstand oder Arbeitsmittel in die zu lenkende Lerntätigkeit einzubeziehen ist und/oder
  - ein geeignetes EDV-Gerätesystem bereits vorhanden ist und stärker für Ausbildungsaufgaben genutzt werden kann.
- Die Lehrprogrammentwicklung sollte von der Auftragserteilung an Autorenkollektive bis zur Erprobung und Freigabe zentralisiert organisiert werden. In die Bestimmung der Themen und Ziele sind die verantwortlichen Fachkommissionen und wissenschaftlichen Beiräte einzubeziehen. Die Lehrprogramme müssen sich in unterschiedliche Lehrkonzeptionen einpassen lassen, der Hochschullehrer sollte sich seine Einsatzvariante ohne größeren Aufwand zusammenstellen können.

- Die Möglichkeiten des Lernenden, den eigenen Lernprozeß mit Hilfe des Systems bewußt und eigenverantwortlich mitzugestalten, müssen in vielfältiger Weise erweitert werden (Zulassung von Aktionen, d. h. Anforderungen und Handlungen aus eigenem Antrieb; Angebot von Varianten usw.).
- Die Lehrprogrammentwicklung ist ihrem Wesen nach ein schöpferischer Prozeß, der nicht allein auf wissenschaftlichen Theorien und Modellen beruht, sondern der praktischen Lehrerfahrung und des pädagogischen Geschicks der Autoren bedarf. Damit entzieht er sich einer durchgängigen Automatisierung, was aber nicht für jeden seiner Teile zutreffen muß. Als eine Möglichkeit zur stärkeren wissenschaftlichen Fundierung der Lehrprogrammentwicklung bietet sich das von OELSCHLEGEL in /5/ beschriebene Verfahren an.

Die experimentelle Einführung von AOS VUZ/AP-64 an der TU Dresden schafft günstige Voraussetzungen für die in enger Kooperation mit dem Moskauer NIIVŠ durchzuführenden Untersuchungen zu effektiven Wirkungsbedingungen und Einsatzvarianten rechnergestützter Lehrsysteme im Ausbildungsprozeß. Der spezifische Beitrag des Forschungszentrums für technische Lehr- und Lernmittel wird vor allem darin bestehen, Beiträge zur Erarbeitung präzisierter pädagogisch-technischer Anforderungen an Lehrsysteme, zur Schaffung einer Methodik der Lehrprogrammentwicklung und zur Ausarbeitung methodischer Einsatzempfehlungen für solche Programme zu liefern. Gleichzeitig sollen erste Schritte zur breiteren Nutzung von Lehrsystemen eingeleitet werden, wozu die Übergabe des Programmpakets AOS VUZ/AP-64 an interessierte Kooperationspartner gehören könnte.

An der TU Dresden soll das Lehrsystem zunächst in drei Lehrgebieten eingesetzt werden. Gemeinsam mit den verantwortlichen Lehrkollektiven wurden dafür folgende Orientierungen festgelegt:

Lehrgebiet Grundlagenmathematik: Entwicklung eines Lehrprogramms für Studenten mit spezifischen Wissens- und Könnenslücken auf der Grundlage vorheriger diagnostischer Untersuchungen. Allmählicher Übergang zum Diagnostizieren während der Lehrprogrammbearbeitung.

Lehrgebiet Informationsverarbeitung im Maschineningenieurwesen: Erarbeitung eines Wiederholungs- und Festigungsprogramms mit hohem Übungsanteil unter gezielter Nutzung

systemspezifischer Mittel zur syntaktischen Analyse von FORTRAN-Elementen. Entwicklung von Makros zur Auswertung bestimmter Aufgabenklassen und Untersuchungen zur motivierenden Funktion des Lehrsystemeinsatzes für die Anwendung rechnergestützter Arbeitsweisen.

Lehrgebiet Russische Sprache: Einsatz eines Übungsprogramms zu grammatikalischen Sachverhalten (Gebrauch der Verbalaspekte) unter besonderer Beachtung differenzierter, lernverlaufsabhängiger Führung und Einschätzung des Lernenden sowie der Gestaltung informativer Protokollübersichten für den Hochschullehrer, Entwicklung und Nutzung von Informationsmöglichkeiten für den Lernenden (Wörterbuch, Übersetzungen, Hilfen usw.) und Ansätzen der eigenverantwortlichen Übungsgestaltung.

Der Einsatz des Lehrsystems AOS VUZ/AP-64 an der TU Dresden ist auch mit einigen Problemen verbunden, deren wichtigste in drei Punkten zusammengefaßt werden können:

- Die vorhandene gerätetechnische Basis (max. 5 Endplätze verfügbar) und deren hoher Auslastungsgrad erzwingen den Systemeinsatz ausschließlich im Selbststudium. Das ist zweifellos ein wesentlicher, aber nicht der allein denkbare und erfolgversprechende Einsatzort.
- Die hohe Dynamik der gerätetechnischen Entwicklung macht langfristig getroffene Entscheidungen für die Perspektive zumindest diskussionswürdig. Wurde vor 3 Jahren zu Recht auf das AP-64-Bildschirmsystem orientiert, steht heute die Frage seines baldigen Verschleißes und der Einführung leistungsfähigerer Mittel, darunter auch von Mikrorechnersystemen.
- Der große Erfahrungsschatz der sowjetischen Hochschule bei der Lehrprogrammentwicklung für AOS VUZ wird aufgrund unterschiedlicher Lehrpläne und -konzeptionen sowie des Zwangs zur Übersetzung nur bedingt nutzbar sein.

Als Schlußfolgerung ergibt sich u. a., die Untersuchungen so anzulegen, daß ihre Ergebnisse weitgehend unabhängig von der konkreten technischen Basis und somit auf beliebige Lehrsystemarten übertragbar sind. Weiterhin wird es erforderlich sein, die aus der Lehrgeräteforschung gewonnenen Erfahrungen in dialektischer Weise aufzuheben und dabei die spezifischen Möglichkeiten rechnergestützter Lehrsysteme gebührend zu beachten.

/1/ 2. Arbeitsseminar DDR-UdSSR zu Problemen rechnergestützter Lehrsysteme vom 3. - 7. 12. 1984 in Dresden und Altenberg: Seminarmaterialien. - 1985. - Dresden, Techn. Univ., Sektion Berufspäd. (Wiss. Beiträge, Reihe Hochschulpäd.) - (im Druck)

/2/ Hartwig, W.-H.; Novikov, V. I.
Das Lehrsystem AOS VUZ - Anleitung für Autoren zur Übersetzung und Eingabe sowie zum Test von Lehrprogrammen. - Dresden und Moskau, Techn. Universität und NIIVŠ 1985 (im Druck)

/3/ Hartwig, W.-H.
Stand, Erfahrungen und Tendenzen der Entwicklung und des Einsatzes von Lehrsystemen im Hochschulwesen der DDR. - 1982. - S. 18 - 30. - Dresden, Pädagogische Hochschule, Sektion Mathematik, WB Informatik. 1. Seminar "Computer als Mittel und Gegenstand der Ausbildung". Seminarberichte, Teil 1

/4/ Hartwig, W.-H.
Das Lehrsystem SPOK-VUZ: Aufbau, Eigenschaften und Anwendungsmöglichkeiten. Ministerium für Hoch- und Fachschulwesen. Schriftenreihe "Informationsverarbeitung im Hoch- und Fachschulwesen". - 1981

/5/ Oelschlegel, J.
Ein Beitrag zur Methodik des Lehrprogrammentwurfs auf der Basis von Tätigkeitsanalysen und Elementen der Entscheidungstheorie. Enthalten in /1/

Dr. Wolf-Henning Hartwig
TU Dresden, Sektion Berufspädagogik
Forschungszentrum für technische Lehr- und Lernmittel
8027 Dresden, Mommsenstr. 13

KÜHNE, TORSTEN

Rationelle Softwareentwicklung als Lehrgegenstand in der Hochschul-Ingenieurausbildung

(Kurzfassung)

Bei der Beschreinbung, Bewertung und Rationalisierung von Problembearbeitungsprozessen typ. Ingenieurtätigkeiten gewinnt die
- Strukturierung und
- Algorithmierung

mittels effektiver Mittel und Methoden der automatengestützten Informationsverarbeitung eine zunehmend entscheidende Bedeutung zur meßbaren Erhöhung der Qualität und des Erneuerungsgrades sowohl materieller als auch geistiger Produkte.
Betrachten wir uns kurz die einzelnen Phasen des ingenieurtechn. Konstruktionsprozesses, die man folgendermaßen definieren kann.

1. Spezifikationsphase (Funktionsfindung)
2. Konzeptionsphase (Prinziperarbeitung)
3. Gestaltungsphase (Gestaltung)
4. Ausarbeitungsphase (Detalllierung)

Anliegen der Automatenstützung des Konstruktionsprozesses ist, möglichst durchgängig alle Phasen im Rechner abzubilden(Einheit von Soft- und Hardware) und die Objekte rechnersystemintern zu manipulieren und zu bewerten.
Der entsprechende Softwareentwurf ist mit einem hohen Maß an Abstraktion verbunden, d.h. die Fähigkeit bei Betrachtung komplexer Systeme und Problemstellungen die wesentlichen von den unwesentlichen Seiten zu trennen. Es entsteht also die Aufgabe, das komplexe Gesamtsystem in hinreichend überschaubare und verifizierbare Teilsysteme und Einzelprinzipien zu zerlegen, wobei die Schritte bei einer solchen
- Dekomposition mit
- Abstraktion

auf unterschiedlichem Niveau verbunden sind.

Aufbereitete technologische Mittel zur Unterstützung dieses Prozesses werden in zunehmender doch bisher in unzureichender(was die

- Auswahl
- Präzisierung
- Bündelung
- durchgängige Anwendbarkeit auf sämtliche Phasen des Konstruktionsprozesses

betrifft) Weise durch die Informatik bereitgestellt.

<u>Technolog. Mittel</u>

<u>Method. Mittel</u>	<u>Progr.techn. Mittel</u>
- Methoden	- Programmsysteme
- Prinzipien	- Programme
- prakt. Vorgehensweisen	- Moduln
- anwendungstechn. Forderungen	- Makros

Der systematische Einsatz der technolog. Mittel ist beim

- Programming in the large

einsichtig und unabdingbar.

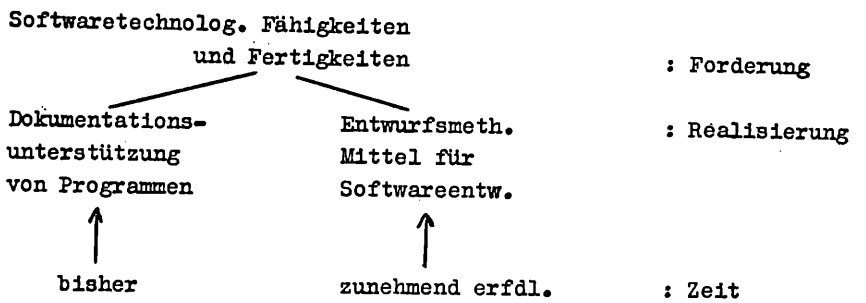
Die Absolventen unseres HS-Ausbildungsprozesses der techn. und ök. Fachrichtungen müssen jedoch in zunehmender Weise zur Beherrschung und Anwendung der Softwaretechnolog. Mittel befähigt werden woraus sich ergibt, an geeigneten Prozessen und Objekten sowie Systemen innerhalb des wiss. geführten Studienprozesses

- Lern- und
- Träinings-

strukturen mit den Entsprech. Inhalten unter Einbeziehung bewährter hochschuldidaktischer Prinzipien zu realisieren. Somit ergibt sich die Aufgabe o.g. technolog. Mittel für den Studienprozess im Umfang des

- Programming in the small

aufzubereiten, wobei über die bisherigen Zielstellungen der vertieften IV-Ausbildung von Ingenieuren damit hinausgegangen wird.



Bei der Erarbeitung der Schwerpunkte bzgl. 'Entwurfsmethod. Lehrinhalte bilden

1. Relevante Inhalte und Relationen der Software-Technologie (Anwendung von Gesetzmäßigkeiten, Methoden und Werkzeuge)
2. Aufgabenklassenbildung bzgl. ihrer Spezifika zu soft.ware-technolog. Komponenten
3. Didaktische Grundpositionen und ergonomische Aspekte

ein Forderungskonzept bzw. Bedingungsgefüge.

Zu 1.

- Strukturierter Entwurf (schrittweise Dekomposition Prozess-Objekt-bezogen)
- Modularer Entwurf (- Funktionsbezogen
  - Systembezogen)
- Prinzip der Konzentrierten Logik (Anwendungsfall für Modul. Entwurf)
- Aussagenlogik (formale Beschreibung von Entscheidungsproblemen und deren Minimierung)

Zu 2.

Vergleichende Untersuchung und Zuordnung von ingenieurtechn-Aufgabenstellungen bzgl. ihrer Dekomposition nach den bekannten Softwareentwicklungsmethoden:

- Jackson - Methode
- Warnier - Methode
- Constantine - Methode
- HIPO } prim. Darstellungsmethoden
- SADT }

Zu 3.

Aus der Realisierung der o.g. Inhalte und der Einbeziehung eines automatengestützten
- Lehrsystemes mit
- interaktiver Arbeitsweise zur
- Lehrprogrammerstellung durch
- den Lehrer (Lehrprogrammautor) und
- Lehrprogrammabarbeitung durch den/die
- Studenten

sowie den gleichzeitigen Einsatz des Rechners als
- Arbeitsinstrument des/der Studenten ergeben sich wirklichkeitsnahe/praxisrelevante Bedingungen zur Ausprägung von
- Selbstständigkeit und Eigenverantwortlichkeit.

Das Automatengestützte Lehrsystem AOS - VUZ(x) wird auf der Rechnerbasis ESER 1022 mit AP 64 an der Hochschule für Architektur und Bauwesen Weimar zur Ausbildung der Studenten der Fachrichtung Bauingenieurwesen eingesetzt werden.

Es bildet die Basiskomponente eines Automatengestützten Trainings- Lehr- und Auskunfts- Systemes A T L A S .

Die in Entwicklung befindlichen Trainings- und Lehrkomponenten werden durch eine
- Methodenbank (Auskunftssystem) mit entspr.
- Nutzerführung ergänzt um den stud. Aneignungsprozess durch Komponenten des
- wiss. Arbeitsprozesses

  sinnvoll zu ergänzen und außerhalb der E-A-Aufgaben auch fachwiss. Forschungsaufgaben durch weitere
- Nutzergruppen wirksam zu unterstützen.

Literatur:

Hartwig,W.H.

(x) Das Lehrsystem SPOK-VUZ ,Fo-Zentrum fürtechn. Lehr-und Lernmittel TUD Schriftenreihe 'Informationsverarbeitung im Hoch- und Fachschulwesen 1981

Verfasser:

Dr.rer.nat. Kühne,T
Hochschule für Architektur und Bauwesen Weimar

Dr. Lemke, Guntram

Praxisorientierte Weiterbildung am Schulungszentrum des VEB Robotron-Anlagenbau, Leipzig.

1. Einleitung

Die effektive Nutzung der vom VEB Kombinat Robotron vertriebenen Erzeugnisse (Hardware) und Leistungen (Software) setzt umfassende Kenntnisse im Bereich der Informationsverarbeitung und der dafür eingesetzten Informationstechnik voraus. Die Hoch- und Fachschulen sowie die Berufsausbildung vermitteln im In- und Ausland das notwendige Grundwissen. Es ist durch spezielle Kenntnisse über die einzusetzenden Anlagen, Geräte, Systeme und Verfahren zu erweitern. Die Schulungseinrichtungen des VEB Kombinat Robotron bieten dafür ein umfassendes Schulungsprogramm. Dieses Schulungsangebot wird ergänzt durch Lehrgänge für eine anlagen- und geräteunabhängige Aus- und Weiterbildung, um zur Deckung des sich bei den unterschiedlichen Bildungsvoraussetzungen ergebenden zusätzlichen Bedarfs beizutragen.

2. Leistungsspektrum und -angebot

Das Schulungsprogramm zeigt ein modular aufgebautes und zwischen den einzelnen Schulungseinrichtungen abgestimmtes Lehrgangssystem. Es ermöglicht den Käufern von Erzeugnissen und Leistungen des VEB Kombinat Robotron sowie den anderen Interessenten die individuelle Gestaltung des Bildungsweges der Mitarbeiter entsprechend den vorhandenen Kenntnissen und Erfahrungen. Besondere Anforderungen des künftigen Tätigkeitsbereiches können ebenfalls berücksichtigt werden. Damit ist ein optimales Verhältnis von Bildungsaufwand und dem zu erreichenden Wissensstand zu erzielen.
Schrittweise konnte unter Beachtung der Anforderungen ein leistungsfähiges, modulares und mit anderen Schulungseinrichtungen des VEB Kombinat Robotron abgestimmtes Lehrgangssystem aufgebaut werden. Es hat sich bereits seit vielen Jahren bewährt und wurde durch ständige Weiterentwicklung den sich verändernden erhöhten Anforderungen angepaßt.

Das Schulungsangebot umfaßt Lehrgänge zur Aus- und Weiterbildung von Spezialisten auf folgenden Gebieten:

- Software
  Systemanalytiker
  Anwendungsprogrammierer
  Systemprogrammierer
  Operatoren
- Hardware
  Anlagentechniker
  Reparaturtechniker

Die Schulungseinrichtungen sind jederzeit bereit, weitere Aus- und Weiterbildungswünsche zu erfüllen, die sich aus den besonderen Anforderungen an die Soft- und Hardware eines Käufers ergeben. Dafür werden individuelle Schulungsabläufe zusammengestellt und Sonderlehrgänge konzipiert.
Die Mitarbeiter der Schulungseinrichtungen für das Angebot von Schulungsleistungen und die Spezialisten der einzelnen Schulungsgebiete geben dafür die notwendige Unterstützung und beraten jeden Interessenten fachgerecht.
Weitere Informationen sind den Schulungsprogrammen zu entnehmen, die auch mehrsprachig zur Verfügung gestellt werden.

3. Schulungsdurchführung

Das Schulungszentrum Leipzig des VEB Robotron-Anlagenbau übt im VEB Kombinat Robotron die Leitfunktion Schulung aus und koordiniert die Arbeit der anderen Schulungseinrichtungen im Kombinat.
Die Umsetzung der Schulungskonzeptionen und des Lehrstoffes erfordert den Einsatz leistungsfähiger Hoch- und Fachschulkader als Lehrkräfte. Die 170 eingesetzten Lehrkräfte sind anerkannte Spezialisten mit einem umfangreichen, durch praktische Arbeit gefestigten Wissen und mit pädagogischen Erfahrungen.

Besondere Anforderungen stellen die in jedem Lehrgang immer wieder auftretenden unterschiedlichen Bildungsvoraussetzungen der einzelnen Lehrgangsteilnehmer und ihre voneinander abweichenden Erwartungen an das Ihnen zu vermittelnde Wissen.

Die Lehrkraft muß daher sofort auf diese Bedingungen angemessen reagieren, ohne das zu erreichende Lehrgangsziel zu gefährden. Das trifft in erhöhtem Maße auf die Lehrgänge mit ausländischen Teilnehmern zu, da die sprachliche Verständigung über Dolmetscher zusätzlichen Aufwand und ein ausgeprägtes Einfühlungsvermögen der Lehrkraft erfordert. Die zusätzliche Sprachqualifizierung und damit die Durchführung von Lehrgängen in Fremdsprachen durch die Lehrkräfte hat bereits zur Verbesserung der Situation ebenso beigetragen wie das viel genutzte Angebot zu Konsultationen nach der eigentlichen Lehrgangszeit und die bereitgestellte umfangreiche fremdsprachige Schulungsdokumentation.

Ein Teil der theoretischen Lehrabschnitte wird auf Wunsch der ausländischen Partner in ihrem Land durchgeführt: Mitarbeiter des Schulungszentrums waren bereits auf allen fünf Kontinenten im Einsatz, haben sich unter oft komplizierten klimatischen sowie anderen landesabhängigen Bedingungen bewährt.

4. Schulungsorganisation

Die inhaltliche Vorbereitung und die Erfüllung der Schulungsaufgaben erfordern:

- Mitwirkung in betrieblichen und überbetrieblichen, beratenden und die Bildungsarbeit koordinierenden Gremien im In- und Ausland
- Beschaffung der für die Erarbeitung der Entwicklungskonzeption des Schulungszentrums benötigten Informationen und Umsetzung der gewonnenen Erkenntnisse auf die Belange des Schulungszentrums
- Lang-, mittel- und kurzfristige qualitative und quantitative Planung des Schulungsbedarfs sowie der anzubietenden und zu erzielenden Schulungsleistungen
- Bilanzierung von Schulungsbedarf, Schulungskapazität und zu erbringenden Schulungsleistungen

- Erarbeitung jährlicher Lehrgangsterminpläne unter Berücksichtigung der zeitlichen Schwankungen im auftretenden Lehrgangsbedarf, der verfügbaren Lehrkräfte, Schulungsräume und -technik
- Angebot der Lehrgänge an die in- und ausländischen Interessenten durch deutsch- und mehrsprachige Informationsschriften, Schulungsprogramme, Kundenberatung im In-und Ausland usw.
- Disposition der Lehrgangsplätze bei Sicherung vorrangig zu erfüllender Schulungsverpflichtungen
- Angebot und Abschluß der ausländischen Lehrgangsteilnehmer während ihres Aufenthaltes in Leipzig
- Berechnung und Analyse der erbrachten Schulungsleistungen.

5. Schlußbemerkungen

Die Vermittlung praxisorientierten und erprobten Wissens war und ist eines der wichtigsten Zeile der Arbeit des Schulungszentrums, ebenso wie die Durchführung umfangreicher Praktika für die Lehrgangsteilnehmer.
Großer Wert wird auf die zeit- und qualitätsgerechte Bereitstellung der Schulungsdokumentation in den benötigten Sprachen gelegt. Das ist oft nur unter hohem persönlichem Einsatz der Lehrkräfte und der technischen Mitarbeiter zu erreichen. Die bisherigen Erfolge rechtfertigen den Aufwand.

Das Schulungsangebot des Schulungszentrums Leipzig umfaßt in Abstimmung mit der Absatzverantwortung des VEB Robotron-Anlagenbau hauptsächlich folgende Schulungsgebiete der Soft- und Hardware:

- Elektronische Datenverarbeitungsanlagen
- Mini- und Mikroechnersysteme, wie Bürocomputer, Basis- und Prozeßrechnersysteme
- Gerätesysteme und Einrichtungen zur Datenfernverarbeitung
- Anwendungslinien und Anwenderlösungen
- Systemunabhängige Lehrgebiete.

Das Lehrgangssystem ist so aufgebaut, daß es der zu erwartenden Entwicklung von Soft- und Hardware entspricht, die Bildungsanforderungen erfüllt, die Qualifizierungszeit verkürzt und den Bildungsaufwand senkt.

Anschrift des Verfassers:

Dr. Lemke

VEB Robotron-Anlagenbau

7010 Leipzig

Gerberstr. 3/5
Schulungszentrum

Mätzel, Klaus

Einsatzergebnisse des Systems PSU in der Ausbildung

Mit dem Einsatz des Systems PSU werden für Aufgaben der Programmentwicklung Möglichkeiten einer solchen sprachlichen Schnittstelle angeboten, die sowohl im Stapel - als auch im interaktiven Betrieb genutzt werden kann und zudem einen einfachen Übergang zu unterschiedlichen Rechnersystemen gestattet.

Aus der Sicht der Ausbildung bietet ein solcher Zugang einen besonderen Vorteil; zum einen können die methodischen Aspekte der problemorientierten Problemlösung unmittelbar auf die Rechnernutzung übertragen werden und andererseits wird die Ausbildung praxisrelevanter, wenn sich Ausbildungs- und Produktionsumgebung nicht unterscheiden.

Prof. Dr. K. Mätzel
Technische Hochschule Karl-Marx-Stadt
Sektion Informatik
9o1o Karl-Marx-Stadt
PSF 964

Klaus Ringhand

PROTOS - Ein Hilfsmittel zum Entwurf rechnergestützter ökonomischer Spiele

Rechnergestützte ökonomische Spiele werden heute überwiegend als Dialogprojekte entwickelt.
Ein Problem beim Entwurf dialogorientierter ökonomischer Spiele ist die Gestaltung der Aus- und Eingaben am Bildschirm.
Hier treffen fachlich-inhaltliche, pädagogisch-methodische und datenverarbeitungstechnische Anforderungen aufeinander. Zur Lösung dieses Problems ist eine gute Kommunikation zwischen den Fachökonomen als geistigem Vater des ökonomischen Spieles und dem Datenverarbeiter als dessen Realisator notwendig. Diese Kommunikation wird durch das Softwarepaket PROTOS unterstützt, indem es die schnelle Entwicklung eines funktionsfähigen Prototyps des zukünftigen rechnergestützten ökonomischen Spieles ermöglicht. Der Prototyp gestattet es, die entworfenen Bilder in der festgelegten Reihenfolge zu betrachten. Dabei sind Eingaben inhaltlicher Art möglich, sie dienen zum Test der Gestaltung der Eingabefelder. Die Eingaben haben keinerlei Wirkung auf andere Bilder oder die Bildfolge. Eingaben zur Bildfolgesteuerung erfolgen durch Steuerzeichenkombinationen.
Das Softwarepaket PROTOS wurde in Zusammenarbeit mit der KMU Budapest in PASCAL entwickelt und auf einem Mikrorechner implementiert.

Dr.-Ing. Alfred Schilling

Konzept der Lehrerweiterbildung in Informatik in Zusammenarbeit mit dem Pädagogischen Kreiskabinett Leipzig-Süd

In der Ausbildung von POS- und EOS-Lehrern spielt die Informatik auch heute noch eine untergeordnete Rolle, da bestenfalls Mathematik-/Physik- und ESP-Lehrer in unterschiedlich geringem Umfang darüber etwas hören. Im IV. Quartal 84 kommen die ersten Heimcomputer der DDR-Produktion in den Handel und mit der wachsenden Zahl solcher Geräte in den Familien werden in zunehmenden Maße auch die Lehrer mit Fragen zum Komplex Informatik konfrontiert werden.
Ab Herbst 1985 werden in den 7. Klassen offiziell Taschenrechner statt Rechenschieber eingeführt. In diesem Zusammenhang wird ein Plan zur Lehrerweiterbildung im Stadtbezirk Leipzig-Süd erarbeitet. Mit einem Kurs zur Einführung der Taschenrechner im Herbst 84 läuft ein Plan zur Weiterbildung nicht nur für Mathematiklehrer an, der im weiteren auch den rationellen Einsatz des Heimcomputers im Unterricht und die dazu notwendigen Grundlagen umfassen wird.

Dr.-Ing. A. Schilling
Karl-Marx-Universität Leipzig, ORZ
7o1o Leipzig
Karl-Marx-Platz

Simon, Ullrich
Neef, Günther

Erfahrungen bei der Ausbildung von Studenten des MIW auf dem Gebiet der Informationsverarbeitung und Mikroelektronikanwendung

1. Einleitung

Die Entwicklung von Wissenschaft und Technik wird insbesondere seit den siebziger Jahren in hohem Maße durch neue Wissensgebiete geprägt, die zu revolutionären Veränderungen von Erzeugnissen, Prozessen und Technologien führen sowie den Arbeitsinhalt von vielen Berufsgruppen zunehmend beeinflussen und verändern. Die Entwicklung, Produktion und vor allem die Anwendung der Mikroelektronik zählen zu den Disziplinen, die sich international in einem historisch kurzen Zeitraum zu Schlüsseltechnologien der Volkswirtschaft profiliert haben. In gleichem Maße dringt die Informatik in viele Bereiche des gesellschaftlichen Lebens ein und führt durch Übertragung von Aufgaben der Informationsverarbeitung auf EDV-Anlagen zu weitrechenden Veränderungen von der Industrie bis in den Dienstleistungssektor hinein.
Vor allem im Maschinenbau, der international mit zu den Schrittmachern für die Anwendung der Informatik und der Mikroelektronik gehört, zeichnet sich langfristig eine durchgreifende Integration von mechanischen und informationellen Prozessen auf der Basis der Mikroelektronik ab. Die durch die bisherige Entwicklung bestätigten Prognosen weisen für den Maschinenbau eine zunehmende Anzahl von rechnergestützten Arbeitsplätzen mit interaktiver Dialogführung, die Einführung von Systemen der durchgängigen Informationsverarbeitung, die bedienerarme Fertigung durch flexible Fertigungssysteme und flexible Montagesysteme mit Industrierobotern mit dem Fernziel des automatisierten, bedienerarmen Betriebes aus. Sowohl die Forschung zu diesen Aufgabenkomplexen als auch ihre zukünftige Nutzung erfordern vor allem von den Hoch- und Fachschulkadern des Maschinenbaues neben einem soliden fachspezifischen Wissen in zunehmendem Maße Fähigkeiten und Fertigkeiten im Umgang mit dem Computer und mit der mikroelektronischen Steuertechnik.
Diese Entwicklungstrends führen zu neuen Anforderungen an die Aus- und Weiterbildung von Absolventen des Maschineningenieurwesens.

2. Anforderungsprofil an das MIW unter dem Aspekt der Informatik und Mikroelektronikanwendung

Ausgehend von den internationalen Entwicklungsrichtungen des Maschinenbaues wurden durch den Beirat für MIW sowie in vielen Fachdiskussionen mit Vertretern der Praxis Anforderungsprofile für den Absolventen des MIW hinsichtlich der Informationsverarbeitung und der Mikroelektronikanwendung erarbeitet. Vordergründig standen hierbei die Zielstellungen, durch die Fähigkeit zum algorithmischen Denken die Einsatzmöglichkeiten und -formen für Mittel und Methoden der Informatik und der Mikroelektronik im Maschinenbau abzuschätzen, mit oder an Computern fachspezifische Aufgaben zu lösen und durch ein erforderliches Grundlagenwissen gemeinsam mit Spezialisten der Informatik und der Automatisierungstechnik interdisziplinär an Automatisierungslösungen für technische Prozesse mitzuarbeiten. Dazu sind den Studenten des MIW unter Beachtung von fachrichtungsspezifischen Anwendungsgebieten Kenntnisse über Gebiete wie z.B.

- Problemanalyse, maschinenbautypische Prozeßbeschreibung
- Konfigurierung und Auswahl von mikroelektronischen Baugruppen und Geräten der Daten- und Steuertechnik
- Programmierung von informationsverarbeitenden Strukturen, Softwaretechnologie, Auswahl und Anpassung von Software
- interaktiv Mensch-Maschine-Dialog usw.

zu vermitteln.

3. Erfahrungen bei der Ausbildung von Studenten

An der Sektion Fertigungsprozeß und Fertigungsmittel der Technischen Hochschule Karl-Marx-Stadt werden Studenten in der Vertiefungsrichtung Fertigungsprozeßgestaltung/Informationsverarbeitung nach der Nomenklatur ASU 4a und in allen Fachrichtungen der Sektion in der Stufe ASU 4b ausgebildet.
Da gegenwärtig die Vermittlung von Kenntnissen zur Informatik und Mikroelektronik im wesentlichen auf die Hochschulausbildung konzentriert ist, müssen größere Anstrengungen zum Erreichen der o.g. Ausbildungsziele unternommen werden. Nach vorausgegangenen Fachdiskussionen in Leitungsgremien der Sektion begann, aufbauend auf dem Wissensstand aus der Lehrveranstaltung "Informationsverarbeitung", die Ausbildung mit einem 30 h umfassenden Intensivkurs über Mikrorechentechnik für die Studenten des MIW.

Zur Darstellung und Aneignung der abstrakten Abläufe in einem Mikrorechner wurde auf Basis des "Polycomputer 880" ein Praktikum aufgebaut. Dazu erhält jede Praktikumsgruppe (2-3 Studenten) eine maschinenbauorientierte Aufgabe, die selbständig bis zum Maschinenprogramm zu bearbeiten und im Praktikum abgearbeitet und getestet wird. In Bild 1 ist der Aufbau eines Praktikumsplatzes dargestellt.

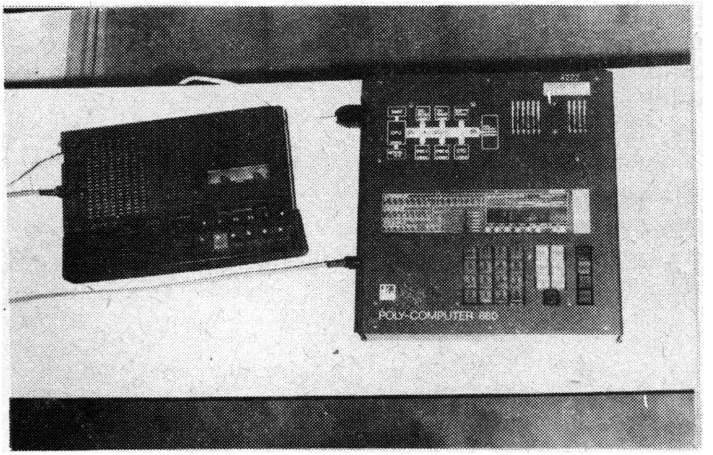


Bild 1: Praktikumsplatz mit Polycomputer 880

Nach Durchführung der Ausbildung für 4 Matrikel wurden in Auswertung der Praktika folgende Erfahrungen abgehoben:

- Die eigenständige Arbeit am Rechner festigte die Kenntnisse und Fähigkeiten über Aufbau, Funktionsweise und Nutzung von Computern.
- Die exemplarische Bearbeitung eines Übungsbeispieles bis auf Assembler- und Maschinensprachniveau verdeutlichte die Programmierungstechnik sowie die Funktion und die Nützlichkeit von problemorientierten Sprachen; für den Maschinenbaustudenten wurde die Informationsverarbeitung "transparent".
- Die Erarbeitung der Programme für die Praktikumsaufgaben erfolgte nach softwaretechnologischen Aspekten in Anlehnung an eine in der Lehrveranstaltung entwickelte Demonstrationsaufgabe. Mit dieser Lösung wurde im 1. Praktikum die Bedienung des Mikrorechners erlernt und durch selbständige schrittweise Abarbeitung die Funktionsweise verdeutlicht.

- Das Interesse und die Bereitschaft zur Mikrorechneranwendung wurde durch die eigene Tätigkeit angeregt.
- Der hohe Aufwand an Praktikumsbetreuung kann nur durch eine entsprechende Zahl von Übungsleitern realisiert werden. Hierzu haben sich ca. 15 Mitarbeiter aus der Sektion qualifiziert.

Aufbauend auf den erreichten Ergebnissen erfolgte ab 1984 eine Erweiterung dieser Ausbildung. Durch eine im 5. und 6. Semester stattfindende Lehrveranstaltung "Grundlagen der Mikroelektronikanwendung im Maschinenbau" werden Kenntnisse und Fähigkeiten über Mikrorechner, Softwaretechnologie, Mikrorechnerarbeitsplätze, Steuerungstechnik für Werkzeugmaschinen, Roboter und Prozeßabläufe, Computergrafik, CAD/CAM-Systeme usw. vermittelt. Dazu sind 60 h Vorlesung, 15 h Seminare und 30 h Praktikum vorgesehen. In Fortführung des Polycomputer-Praktikums werden gegenwärtig in analoger Form

- ein Praktikum zur bildschirmorientierten Arbeit an einem Heim- bzw. Bürocomputer
- ein Praktikum an einer CNC-Steuerung (siehe Bild 2) und
- ein Praktikum an einer Industrierobotersteuerung (siehe Bild 3)

vorbereitet.

Bild 2: CNC-Steuerung

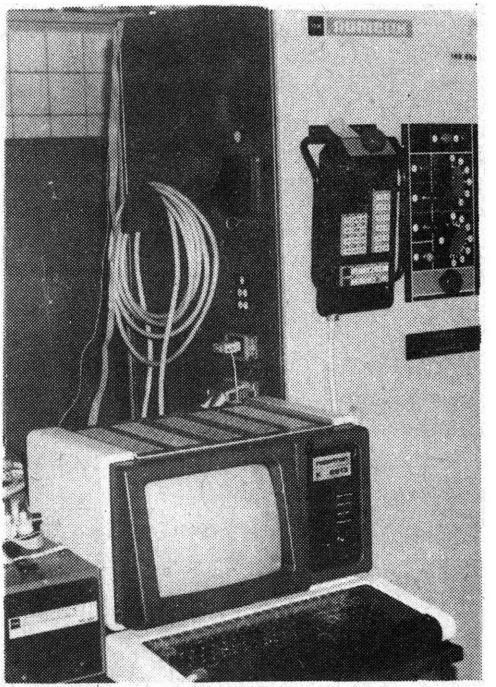


Bild 3: Industrieroboterstcuerung

Zur Gewährleistung der erforderlichen Sicherheit arbeiten die Steuerungen CNC-H 645 und IRS 650 im Simulationsbetrieb. Für Praktikumsgruppen mit syntaktisch fehlerfreien Programmen besteht die Möglichkeit der Abarbeitung ihrer Lösung auf Steuerungen mit angeschlossener Werkzeugmaschine bzw. mit funktionsfähigem Roboter.

Verfasser: Prof.Dr.sc.techn. U. Simon, TH Karl-Marx-Stadt
Sektion Fertigungsprozeß und Fertigungsmittel
9010 Karl-Marx-Stadt, PSF 964
Dr.-Ing. G. Neef, TH Karl-Marx-Stadt
Sektion Fertigungsprozeß und Fertigungsmittel
9010 Karl-Marx-Stadt, PSF 964.

Wagner, Klaus

Weiterbildung auf dem Gebiet der Informationsverarbeitung: Ausgangssituation sowie Ziele, Erfahrungen und Probleme eines Weiterbildungszentrums

1. Einleitung

Als wesentliche Prämisse für den technologischen Fortschritt in einem Lande wird angesehen, daß die Zuwachsrate des Lernens größer sein muß als die Zuwachsrate des technologischen Wandels. Der rasche informationstechnologische Wandel, das Vordringen der automatisierten Informationsverarbeitung (AIV) in immer neue Anwendungsbereiche, sowie neue Prinzipien, Methoden und Techniken der Rationalisierung der geistigen Arbeit erfordern einen breiten Bildungsstand, der durch gezielte Weiterbildung zu erhalten und auszubauen ist. Oberstes Ziel eines solchen Lernverständnisses muß es sein, bei allen Werktätigen eine allgemeine Aufgeschlossenheit und positive Grundeinstellung für die anstehenden technologischen Veränderungen, die eine Umstellung von Organisationsstrukturen, Arbeitsinhalten und Arbeitsabläufen bewirken, zu vermitteln. Für Hoch- und Fachschulkader ist es überdies notwendig, die AIV, letzthin den Computer, als Problemlösungsinstrument zu verwenden.

Aus- und Weiterbildung an den Universitäten und Hochschulen unserer Republik sind auf die Bewältigung der anstehenden informationstechnologischen Veränderungen ausgerichtet, genügen jedoch noch nicht in allen Positionen den künftigen Anforderungen. Gegenstand vorliegenden Beitrages ist die Weiterbildung. Ausbildungsaspekte werden nur am Rande berührt. Ziel des Beitrages ist die Darlegung der Ausgangssituation, wie sie für den schrittweisen Aufbau eines Weiterbildungszentrums für Informationsverarbeitung von Belang ist, sowie die Darlegung der Ziele, Erfahrungen und Probleme beim Aufbau eines solchen Weiterbildungszentrums an der Humboldt-Universität zu Berlin.
Die Ausführungen stützen sich teilweise auf einen Beitrag zur INFO 84 /1/.

2. Ausgangssituation

Nach einer Untersuchung der Gesellschaft für Informatik aus dem Jahre 1982 werden bis 1990 in der BRD 65 % aller Beschäftigten AIV-Wissen benötigen /2/. Der Akzent der modernen Industriearbeit verlagert sich von der physischen Anstrengung hin zur Verarbeitung von abstrakten Symbolen. Eine weitere Untersuchung dieser Gesellschaft (Mai 83) verdeutlichte, daß die Mehrheit der BRD-Bevölkerung eine ambivalente Einstellung zum Computer besitzt, wobei jedoch Personen mit Arbeitserfahrungen mit dem Computer diesen in seinen Auswirkungen generell positiver beurteilen als Personen ohne solche Erfahrungen /3/.

Es ist hier weniger der Raum zu diskutieren, inwieweit diese Auswirkungen für uns von Bedeutung sind und in welchem Tempo sich der informationstechnologische Trend, insbesondere die Verschmelzung von Datenverarbeitungs-, Textverarbeitungs- und Kommunikationstechnologie zu integrierten Informations- und Kommunikationssystemen, in unserer Republik vollzieht. Vielmehr ist es erforderlich zu erkennen, daß die daraus resultierenden Anforderungen an die Aus- und Weiterbildung auf uns zukommen und daß auch unter sozialistischen Verhältnissen, die soziale Sicherheit als ein weltweit beachtetes Kriterium gesellschaftlicher Überlegenheit garantieren, subjektives Beharrungsvermögen und zahlreiche Innovationswiderstände überwunden werden müssen.

Die Rolle des subjektiven Faktors nimmt zu. In einer Gesellschaft, in der die Entwicklungsprozesse bewußt und zielgerichtet reguliert werden, können gerade von diesem Faktor Impulse ausgehen, die die (informationstechnologische) Entwicklung beschleunigen und in die erforderliche Bahn lenken oder auch - kraft der Traditionen und Gewohnheiten der Werktätigen - bremsen. Das Wissen um die neuen Technologien stellt ein bedeutendes Potential zur Gewährleistung ihrer Beherrschung und Akzeptanz dar. Angehörige hochqualifizierter Berufsgruppen, die heute nicht im entferntesten daran denken, daß Computer einen Teil ihrer Arbeit bewältigen können (z.B. Ärzte, Juristen, Lehrer) werden förmlich überfahren, wenn ihnen in den nächsten Jahren Ex-

pertensysteme (der 5. Rechnergeneration) als "Wissensverstärker" gegenüberstehen. Die Zukunft der AIV hat bereits begonnen, und es ist Aufgabe der universitären (Aus- und) Weiterbildung, den notwendigen Bildungsvorlauf zu schaffen. Diesbezügliche Anstrengungen an den Bildungseinrichtungen der imperialistischen Hochburgen sprechen für sich (vgl. z.B. /4/).

Für die in der AIV tätigen (und weiterzubildenden) Hoch- und Fachschulkader unserer Republik stellt sich die gegenwärtige Situation in komprimierter Form wie folgt dar:

1. Die Mehrheit der in der AIV tätigen Spezialisten (Informatiker) besitzt keine solide Grundausbildung in der AIV. Es handelt sich hier um Ausbildungsversäumnisse der 60er und frühen 70er Jahre.

   Irgendeine artfremde Ausbildung und oft nur kurze Berufserfahrung, aber offensichtliche Programmierbegabung, schienen ausreichend zu sein, um nach einer zweiwöchigen Ausbildung zum Erlernen einer Programmiersprache und nach vierzehntägigem Erwerb von Grundkenntnissen eines Betriebssystems Software-Entwicklung betreiben zu können.

2. Mit der massenhaften Verbreitung von Mikroprozessoren, Büro- und Personalcomputern werden AIV-Kenntnisse unterschiedlichen Grades bei Nichtinformatikern benötigt. Bereits gegenwärtig zeichnet sich ein Mangel bei "Bindestrich-Informatikern" ab, das sind Fachspezialisten (Ingenieure, Betriebswirtschaftler, Mediziner usw.) mit umfangreichen Kenntnissen der AIV bzw. Informatik. Es zeigt sich immer deutlicher:

   Die Praxis benötigt vorrangig den Fachspezialisten mit Informatikkenntnissen und weniger den reinen Informatiker mit theoretisch-abstrakter Spezialisierung (vgl. hierzu auch /5/).

   In den Betrieben und Einrichtungen fehlt es insbesondere an Praktikern zur Entwicklung rechnergestützter betrieblicher Informationssysteme. Fachwissen hat hier Vorrang vor Informatikwissen. Die Integration von Informatik-Inhalten in die technische, naturwissenschaftliche und wirtschaftswissenschaftliche Aus- und Weiterbildung wird insofern immer mehr zu einem Schwerpunktproblem. Während bei einer Reihe von Ausbildungsrichtungen die sog. Endbenutzerschulung (Handhabung von Standardanwendungsprogrammen und fachspezifischen Softwarewerkzeugen, z.B. Methodenbanken) ausreichend ist, wäre z.B. für betriebliche Informationssysteme ein Studium Sozia-

listische Betriebswirtschaft mit Studienschwerpunkt Informatik wünschenswert. Wohlgemerkt, von Informatik - als Wissenschaft von der Struktur, den Prinzipien, Methoden und Mitteln der AIV, unter besonderer Berücksichtigung des Organisationsaspektes - ist hier die Rede, nicht von irgendeiner Programmierausbildung und Vermittlung von Gerätebeschreibungen ("Struktur- bzw. Systemwissen statt Faktenwissen" /1/).

3. Während in den Betrieben und Kombinaten die AIV ständig neue Anwendungsgebiete erschließt und günstige rechentechnische Voraussetzungen bestehen (Gleiches gilt für einige Weiterbildungseinrichtungen), können im MHF-Bereich die rechentechnischen Voraussetzungen für eine hocheffektive Aus- und Weiterbildung sowie (Grundlagen-) Forschung auf dem Gebiet der Informationsverarbeitung nicht immer voll befriedigen. Da universitäre Weiterbildung vorrangig mit Vermittlung neuester wissenschaftlicher Erkenntnisse zu tun hat und dabei zwangsläufig auf die aus der Grundlagenforschung abgehobenen Ergebnisse mit applikativem Charakter zurückgreifen muß, ergeben sich ernsthafte Probleme bei der Gewährleistung des Bildungsvorlaufs durch die Hochschulforschung.

3. Ziele, Erfahrungen und Probleme beim schrittweisen Aufbau eines Weiterbildungszentrums für Informationsverarbeitung

Z i e l e :

Durch Ministerweisung übernahm die Humboldt-Universität zu Berlin ab Sept. 1982 die Funktion einer territorialen Leiteinrichtung für die Weiterbildung auf dem Gebiet der Informationsverarbeitung. Gleichzeitig wurde am Organisations- und Rechenzentrum der Humboldt-Universität mit dem schrittweisen Aufbau eines Weiterbildungszentrums für Informationsverarbeitung begonnen. Damit soll ein wirksamer Beitrag zur raschen Verbreitung neuester Erkenntnisse geleistet werden.

Die Weiterbildung wird von mehreren Struktureinheiten der Humboldt-Universität - unter Federführung der ORZ - getragen und von weiteren wissenschaftlichen Einrichtungen der Hauptstadt unterstützt. Das Weiterbildungsprogramm wird im Juni eines jeden

Jahres an die interessierten Betriebe und Einrichtungen versendet. Das Programm überstreicht folgende Gebiete:

- Grundlagen der AIV
- Gestaltung automatisierter Informationssysteme
- Anwendung und Entwicklung von Betriebssystemkomponenten
- Moderne Programmiersprachen, -umgebungen, -werkzeuge und -methoden
- Dialogsysteme
- Verfahrensorientierte Programmsysteme
- Aufbau und Nutzung von Datenbanken
- Nutzung ausgewählter Systeme der Klein- und Mikrorechentechnik
- Rechnergestützte Arbeitsplätze für ausgewählte Aufgabentypen
- Grafische Datenverarbeitung
- Datenkommunikation

Neben der raschen Verbreitung praxisrelevanten AIV-Wissens (z. B. Nutzung funktions- und anwendungsorientierter Software) und neuer Entwicklungsrichtungen (z.B. "Büroautomatisierung", logische Programmierung) soll künftig die Vermittlung von mehr Theorie eine Schlüsselstellung bei der Arbeit des Weiterbildungszentrums einnehmen. Oft ist es gerade der Mangel an Theorie, der die in sich hochdifferenzierte Praxis der AIV so schwer verstehbar und überschaubar macht sowie die notwendige Kommunikation und Kooperation erschwert /1/. Für die universitäre Weiterbildung eröffnet sich hier ein breites Betätigungsfeld.

Dominierende Organisationsform der Weiterbildung ist der Lehrgang (einwöchig, zweiwöchig; meist aber ein Tag/Woche, sich über mehrere Wochen erstreckend). Mit Beginn des Jahres 1985 erscheint zur Unterstützung der Weiterbildung eine Heftreihe "Informatik-Skripten" in loser Folge.

E r f a h r u n g e n :

Während im Studienjahr 1982/83 lediglich vier Lehrgänge mit insgesamt 138 Stunden und 105 Teilnehmern realisiert werden konnten, sind es im Studienjahr 1984/85 bereits 26 (angebotene) Lehrgänge mit insgesamt 525 Stunden und 510 (angemeldeten) Teilnehmern. 15 % der Teilnehmer sind promovierte Kader.

Begehrt sind die Lehrgänge für neue Programmiersprachen, Softwaretechnologie und vor allem Büroautomatisierung/Bürocomputer.

Bei letzteren konnten nicht alle Anmeldungen berücksichtigt werden. Gleichwohl muß hierzu festgestellt werden, daß die Ausbildung/Umschulung für neue Hard- und Softwaresysteme, inklusive Programmiersprachen (z.B. BASIC), nicht primäres Ziel der universitären Weiterbildung sein kann, zumal bis 1990 nur 4 % des Hoch- und Fachschulkaderbestandes der DDR an den Universitäten und Hochschulen weitergebildet werden können /6/. Sofern Programmiersprach-Lehrgänge durchgeführt werden, geht es hier primär um die methodischen Aspekte einer Sprache (also nicht um "programming in language", sondern um "programming into language").

P r o b l e m e :

Den Hauptanteil der Weiterbildung tragen gegenwärtig das ORZ der Humboldt-Universität und das ORZ der Hochschule für Ökonomie. Überwiegend wissenschaftliche Mitarbeiter - und nicht Hochschullehrer - fungieren als Lehrkräfte. Dieser Zustand kann nicht befriedigen. Indes: Die Einbeziehung der die Weiterbildung mittragenden Sektionen bereitet Schwierigkeiten, auch wenn die dort vertretenen Lehrgebiete in unmittelbarer Nachbarschaft der Informatik angesiedelt sind. Zwar bietet die Humboldt-Universität mit ihrem breitgefächerten Wissenschaftsspektrum nahezu ideale Voraussetzungen für eine breitangelegte, anwendungsorientierte Informatik-Weiterbildung (im Sinne einer "Bindestrich-Informatik"), jedoch ist interdisziplinäre Weiterbildung eben mehr als die bloße Aneinanderreihung von Weiterbildungsmaßnahmen verschiedener Wissenschaftsdisziplinen. Auch fehlt es z.Z. noch an interdisziplinärer Forschung, die für die Lösung sowohl komplexer wissenschaftlicher Probleme der Grundlagenforschung als auch der angewandten Forschung geeignet ist. Der Zusammenhang zwischen wissenschaftlicher Forschung, insbesondere anwendungsorientierter Grundlagenforschung, und Weiterbildung muß noch stärker als bisher erkannt werden. Nicht zuletzt erweist sich Weiterbildungstätigkeit als besonders forschungsfördernd.

4. Schlußbemerkungen

Kontinuierliche Weiterbildung ist notwendig für die Beherrschung und Akzeptanz der neuen Technologien. Die Humboldt-Universität kann neben umfassenden Weiterbildungsaktivitäten auf dem Gebiet der Mikroelektronik (vgl. /6/) nunmehr auch auf erste Erfahrungen bei der Weiterbildung auf dem Gebiet der Informationsverarbeitung verweisen. Damit wurde ein wesentlicher Beitrag zur raschen Verbreitung praxisrelevanter Kenntnisse einer Schlüsseltechnologie geleistet, zumal die Anwendung der Mikroelektronik primär vermittels Informationsverarbeitung erfolgt (Anschlußtechnologie). Insgesamt bedarf es jedoch noch erheblicher Anstrengungen, um die Weiterbildung in Zukunft gleichrangig neben die Aufgaben in der Ausbildung und in der Forschung zu setzen. Weitere Anstrengungen sind notwendig bei der Integration von Informatikinhalten in die Fachausbildung.

Literatur

/1/ H. Lehmann, R. Tschirschwitz, K. Wagner: Einige Erfahrungen und Schlußfolgerungen zur Weiterbildung auf dem Gebiet der Informationsverarbeitung; in: INFO 84, Dresden 6.-10.2. 1984, Bd. 2, S. 158-162.

/2/ J. Twiehaus: Berufsbilder und Anforderungen in der Datenverarbeitung im Wandel; in: Office Management 7-8/1984, S. 659-661.

/3/ K. Lange: Das Image des Computers in der Bevölkerung; in: GMD-Spiegel 2/1984, S. 52.

/4/ H.R. Hansen: Mikrocomputer in der US-amerikanischen Hochschulausbildung; in: Angewandte Informatik 26(1984)11, S. 441-451.

/5/ H. Müller-Merbach: Informatikausbildung für die Zukunft; in: Angewandte Informatik 26(1984)2, S. 66-67.

/6/ B.P. Nerlich, K. Albrecht: Weiterbildung an Universitäten und Hochschulen - Zur postgradualen Bildung von Hochschulkadern an der Humboldt-Universität zu Berlin; in: Das Hochschulwesen 32(1984)12, S. 328-330.

Verfasser: Dr. Klaus Wagner
Beauftragter für Weiterbildung
Organisations- und Rechenzentrum der Humboldt-Universität zu Berlin
1086 Berlin, PSF 1297

SEKTION 3
Relationale Datenbasen

Aden, Horst

Abschätzung der erforderlichen Anzahl von Direktzugriffen zur Komplettierung von Daten

1. Einleitung

Diesem Beitrag liegen Vorbetrachtungen zur Entwicklung des Listgenerators für das Programmsystem VIV aus dem Jahre 1973 zugrunde. Da es bei Anwendern, Projektanten, sogar Programmierern oft am Verständnis für die aus den Anforderungen resultierenden Hauptspeicher- und Zugriffsproblemen mangelt, sollen diese Betrachtungen doch noch veröffentlicht werden. Sie sind sicher auch für Datenbankaufgaben von Interesse.

2. Abschätzung der zur Komplettierung einer Hauptgruppe erforderlichen Sätze

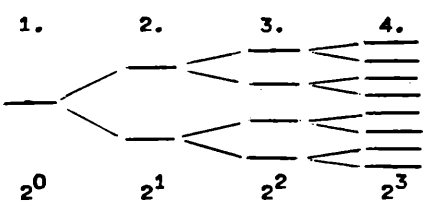
Geht man von der Annahme aus, daß Datensätze mit N relevanten Ordnungsbegriffen auszugeben sind und jede Gruppe die gleiche Anzahl von a Untergruppen enthält, so ergibt sich die Gesamtzahl s unterschiedlicher Schlüssel einer Hauptgruppe zu:

$$s = \sum_{n=1}^{N} a^{n-1} = \frac{a^N - 1}{a - 1} \qquad (2.1)$$

Die Anzahl der Ergänzungssätze e, zum Beispiel Texte aus einer Nomenklatur, die zu jedem der a^{N-1} Datensätze einer Gruppe zu beschaffen sind, beträgt damit

$$2 > e = \frac{a^N - 1}{(a-1)\cdot a^{N-1}} \approx \frac{a}{a - 1} > 1 \quad \text{für } a > 1 \qquad (2.2)$$

Mit a = 2 ergibt sich der ungünstigste Fall, bei dem jede Gruppe nur zwei Untergruppen bzw. Sätze enthält. Vier Ordnungsbegriffe und a = 2 führen zu folgender Schlüsselhierarchie:



Ordnungsbegriffsstufe

8 Sätze (A2.1)

Schlüssel

Die Anzahl der je Datensatz erforderlichen Ergänzungssätze beträgt für dieses Beispiel 15/8.
Folgende Tabelle verdeutlicht die Abhängigkeit der Anzahl von Ergänzungssätzen e je Datensatz in Abhängigkeit von der Anzahl der Sätze je Gruppe a für große N:

a	2	3	4	5	6	10
e	2	1,5	1,33	1,25	1,2	1,1

(T2.1)

Wie allgemein bekannt und aus der Darstellung (A2.1) ersichtlich ist, wird die überwiegende Anzahl von Ergänzungssätzen auf der niedrigsten Ordnungsbegriffsstufe gebraucht. Allgemein ist der Anteil der Ergänzungssätze aus der n. Ordnungsbegriffsstufe:

$$en \approx (a - 1)a^{-(N - n + 1)} \qquad \text{mit } a^N \gg 1 \qquad (2.3)$$

Die Abhängigkeit der erforderlichen Ergänzungssätze für die drei niedrigsten von fünf Ordnungsbegriffsstufen und verschiedene Anzahlen von Sätzen bzw. Untergruppen zeigt die Tabelle 2.2.

en
(N = 5)

a \ n	3	4	5
2	0,125	0,25	0,5
3	0,07	0,22	0,67
4	0,09	0,19	0,75

(T2.2)

3. Abschätzung der zur Komplettierung bezüglich eines Ordnungsbegriffs erforderlichen Zugriffe

Da die Ergänzungssätze gewöhnlich geblockt sind, ist zu erwarten, daß nicht für jeden dieser Sätze ein Zugriff erforderlich ist. Die Wahrscheinlichkeit dafür, daß mehrere unmittelbar benötigte Ergänzungssätze in einem Block gefunden werden, so daß nicht jeder einzeln geladen zu werden braucht, hängt von der Anzahl der Blöcke je Nomenklatur b, der Übereinstimmung der Sortierungen der Daten- und Ergänzungssätze nach ihren Schlüsseln und der Verteilung der geforderten Ergänzungssätze über die Nomenklatur ab.

Im folgenden setzen wir eine Gleichverteilung voraus. Die Wahrscheinlichkeit zwei Ergänzungssätze in einem Block zu finden, ergibt sich bei unterschiedlicher Sortierung zu:

$$W2U = \frac{1}{b} \tag{3.1}$$

Die Wahrscheinlichkeit zwei Ergänzungssätze, die wie die Schlüssel in den Datensätzen sortiert sind, in einem Block zu finden, läßt sich folgendermaßen ableiten. Befindet sich der erste zweier Ergänzungssätze in einem Block, so kann sich der zweite nur im selben oder einem der folgenden Blöcke befinden. Die Gesamtanzahl der Fälle ergibt sich damit zu

$$\sum_{n=1}^{b} n = \frac{b(b+1)}{2} \tag{3.2}$$

von denen b Fälle zwei Sätzen in einem Block entsprechen. Damit ergibt sich die Wahrscheinlichkeit dafür, daß zwei Ergänzungssätze bei gleicher Sortierung in einem Block gefunden werden, zu

$$W2S = \frac{2}{b+1} \tag{3.3}$$

worin $b \geqslant 1$ die Anzahl der Blöcke der Nomenklatur ist.
Folgende Tabelle verdeutlicht die Abhängigkeit der Wahrscheinlichkeiten W2U und W2S von der Anzahl b der Blöcke einer Nomenklatur.

b	1	2	3	4	5	7	9	19
W2U	1	0,5	0,33	0,25	0,2	0,14	0,11	0,05
W2S	1	0,67	0,5	0,4	0,33	0,25	0,2	0,1

(T3.1)

4. Zusammenfassung und Schlußfolgerungen

Während man bei der rein sequentiellen Datenverarbeitung mit zwei bis drei Dateien und Blockungsfaktoren von zehn mit etwa 0,2 bis 0,3 Zugriffen pro Ergebnissatz auskommt, vervielfacht sich die Anzahl der Zugriffe, falls nichtsequentielle Komplettierungen erforderlich werden, bis auf Werte von 1 bis 2 Zugriffen je Datensatz. Dementsprechend erhöhen sich die Lauf-

zeiten. Diese krasse Vervielfachung der Zugriffe wird in der Praxis auf Grund der ungleichen Verteilung der geforderten Ergänzungssätze, die effektiv einer Reduzierung der Nomenklaturblockanzahl gleichkommt, selten erreicht. Der Sachverhalt macht trotzdem zugriffsreduzierende Maßnahmen erforderlich. Dazu zählen:

- Reduzierung der Anzahlen der Blöcke der Nomenklaturen, insbesondere der zu erwartenden rangniedrigsten durch komprimierte Informationen wie zum Beispiel kurze Texte und Bitvektoren
- vollständiges oder selektives Laden der Ergänzungssätze zum rangniedrigsten Ordnungsbegriff in den Hauptspeicher ($\hat{=}$ b = 1)
- vollständiges oder selektives Laden der Ergänzungssätze aus kleinen Nomenklaturen zu möglichst rangniedrigen Ordnungsbegriffen
- Kompression der benötigten Ergänzungssätze auf Arbeitsspeichern (Reduzierung von b)
- Beachtung der Sortierung (kleiner Effekt)
- Anlage von zwei Puffern zumindest für die rangniedrigeren Ordnungsbegriffe und geordnete Eingabe der Nomenklaturblöcke im voraus mit einem Minimum an Armbewegungen (gepufferte, gekettete Eingabe)
- Wenn es die Aufgabenstellung erlaubt, sollten die umfangreichsten Ordnungsbegriffe den höchsten Sortierrang in den Datensätzen erhalten.

Mit dem erwähnten Listgenerator wurden in 100 k HS etwa 0,65 Direktzugriffe zu Ergänzungsinformationen pro in den Listen verwendeten Datensatz erreicht.

Dr. Buchholz, Bernd

PILOT - eine Mehrtupelschnittstelle für das Datenbankbetriebssystem TOPAS

1. Quellen

Die Datenmanipulationssprache PILOT (programming interface language for TOPAS) wird als universelle nutzerfreundliche Mehrtupelschnittstelle angesehen.
Falkenberg (/1/) stellt für die Entwicklung künftiger Datenmanipulationssprachen folgende Forderung auf:
"Erstrebenswert ist, daß diejenigen Regeln, die häufig auftreten, deskriptiv formuliert werden können. Andererseits ist es aber wegen der großen Vielfalt möglicher Regeln unzweckmäßig eine rein deskriptive Sprache vorzunehmen. Die Einführung prozeduraler Elemente verhindert einerseits, daß die Sprache zu unhandlich wird und garantiert andererseits die Formulierbarkeit auch beliebig komplexer und ausgefallener Regeln".
Als die am häufigsten in Datenmanipulationssprachen auftretenden Regeln sind Anweisungen zur Manipulation der Daten in der Datenbank anzusehen. Es wurden deshalb zur Gestaltung der Syntax der Mehrtupelschnittstelle Konstruktionen der Datenmanilulationssprache SEQUEL (/2/) herangezogen.
SEQUEL wird zu den abbildungsorientierten Datenmanipulationssprachen gezählt. Besonderer Wert wurde beim Entwurf dieser Sprache auf leichte Handhabbarkeit, gute Strukturierung und hohe Nutzerfreundlichkeit gelegt.
SEQUEL, als relational vollständige Sprache, ist sowohl eigenständig mit recht großen Leistungsumfang, als auch in ein PL1-Programm eingebettet, nutzbar.
Um den Nutzer von der Erlernung der Programmiersprache PL1 zu befreien, wurde in der Sprache PILOT auf den vollständigen Leistungsumfang von PL1 bewußt verzichtet und dafür eine Auswahl von Steuerelementen problemorientierter Programmiersprachen in der Notation der Entwurfssprache "Pseudocode" (/3/) aufgenommen.
Da in PILOT keine Bezugnahmen auf die konkrete Datenbank existieren, ist PILOT eine allgemein nutzbare Sprache auf Mehrtupelbasis für das Datenbankbetriebssystem TOPAS.

PILOT enthält deskriptive Sprachanweisungen zur Datenmanipulation, zur Recherche, zur Ein- und Ausgabe von Variablen und zum Anschluß von Nutzerprozeduren sowie prozedurale Sprachelemente zur Programmsteuerung.
Der Nutzer dieser Sprache ist von der Kenntnis der Zugriffshilfen und der konkreten Datenformate der Attribute der Relationen befreit (/4/).
Die Implementierung erfolgte mittels des Dresdner Programmtransformationssystems DEPOT (/5/).

2. Formulierung einfacher Retrieval-Anforderungen

Aufgrund der Konstruktion der Sprache PILOT kann sich der Nutzer die Sprachelemente stufenweise aneignen. Bereits mit minimalen Kenntnissen sind umfangreiche einfache Retrieval-Anforderungen leicht und effektiv formulierbar.
Zur Demonstration seien zunächst zwei Relationen mit folgendem Aufbau eingeführt:

Relationsname: STAM
Attribute: AKTNR (Aktennummer)
ANMDAT (Anmeldedatum)
PATART (Patentart)
Relationsname: ERFD
Attribute: EAKTNR (Aktennummer)
ENAME (Erfindername)
GESCHL (Geschlecht)
GEBDAT (Geburtsdatum)

Grundlage der Formulierung von Retrieval-Anforderungen bilden die rekursiv anwendbare FIND-Anweisung und die Ausgabeanweisung PUT.
Beispiel: Ermittlung der weiblichen Erfinder, die jünger als 30 Jahre sind und nach dem 01.01.1980 ein Ausschließungspatent angemeldet haben.

```
FIND AKTNR
FROM STAM
WHERE ANMDAT > 800101 AND PATART='A'
  FIND ENAME
  FROM ERFD
  WHERE EAKTNR=AKTNR AND GESCHL='W'
        AND GEBDAT < 07.10.54
    PUT ENAME
  ENDFIND
ENDFIND
```

3. Datenmanipulationsmöglichkeiten

Manipulationen innerhalb der Relationen sind mittels PILOT in vielfältiger Weise möglich. Tupel können eingefügt oder gelöscht, Attributausprägungen verändert, Relationen um Attribute erweitert oder reduziert sowie Relationen angelegt und aufgegeben werden.

Beispiele: a) Füge ein Tupel mit der Aktennummer 1234567 und dem Anmeldedatum 6.12.83 in die Relation STAM ein.

```
INSERT AKTNR=1234567, ANMDAT=831206
INTO STAM
```

(Bem.: Die Ausprägung des Attributes PATART in diesem Tupel wird mit 'B' initialisiert. Das Anmeldedatum ist auch in der Form 06.12.83 formulierbar.)

b) Ändere die Patentart der Erfindung mit der Aktennummer 1234567 auf "Wirtschaftspatent" ('W').

```
UPDATE PATART='W'
IN STAM
WHERE AKTNR=1234567.
```

c) Erweitere die Relation STAM um die Attribute "Patentschriftnummer", und "Seitenzahl der Patentschrift" (SEITEN).

```
EXPAND STAM:
  PSNR LIKE AKTNR
  SEITEN INT (3)
```

4. Ein- und Ausgabeanweisungen

Mittels der in PILOT enthaltenen Ein- und Ausgabeanweisungen sind die Verarbeitung sequentieller Dateien, die Erstellung von Druck- bzw. Bildschirmausgaben und die Eingabe von Verarbeitungsdaten (Parameter) während der Programmausführung möglich. Somit ist PILOT u.a. sowohl zur Erstellung von Aktualisierungsprogrammen für Relationen, für die Herstellung sequentieller Auszüge aus der Datenbank zur Weiterverarbeitung mittels eines Tabellengenerators oder in anderen Projekten, als auch die Erzeugung beliebig komplizierter Listen nutzbar.

Beispiel: Ermittle die Aktennummern derjenigen Erfindungen, die in vorzugebenden Berichtszeiträumen angemeldet wurden.

```
DECLARE BZR INT (6)
        BIS INT (6)
ENDDCL;
WHILE EVEN DO
   GET BZR,BIS;
   FIND AKTNR
   FROM STAM
   WHERE ANMDAT ≥ BZR AND ANMDAT ≤ BIS
      PRINT SKIP,'AKTNR=',X(1),AKTNR
   ENDFIND
ENDWHILE
```

(Bem.: Die PRINT-Anweisung kann durch PUT AKTNR ersetzt werden, da damit das gleiche Druckbild erzeugt wird. Die geforderte Eingabe (SYSIN) wird durch die SYSPRINT-Ausgabe "PLEASE,GET BZR,BIS" signalisiert. Die Eingabe hat die Form: z.B.: BZR=840101, BIS=841007;)

5. Sonstige Anweisungen

PILOT enthält eine Reihe von prozeduralen Anweisungen zur Projektsteuerung, deren Semantik mit den entsprechenden Anweisungen in problemorientierten Programmiersprachen vergleichbar ist. Zur Erhöhung der Effektivität der PILOT-Programmierung wurden eine Reihe von Builtin- Funktionen implementiert.

Insbesondere zur Unterstützung der PL1- Anwendungsprogrammierer wurden Möglichkeiten geschaffen, PL1- Quelltext einzufügen sowie PL1- Prozeduren aus PILOT- Programmen aufzurufen.
PILOT enthält eine Datenschutzfunktion auf Attributebene.
Während der Abarbeitung von PILOT- Programmen entsteht ein Laufprotokoll sowie Statistiken der angeschlossenen Relationen. Im Falle von E/A- Fehlern endet das PILOT- Programm mittels spezieller USER-ABEND-Codes. Somit sind PILOT- Programme auch unter Jobnetzsteuerung nutzbar.

6. Systemtechnische Voraussetzungen

Es gelten die zur Nutzung des Datenbankbetriebssystems TOPAS einzuhaltenden Voraussetzungen (/6/). Zur Übersetzung eines PILOT- Programmes ist ein Hauptspeicherbereich von 200 K Bytes erforderlich.

7. Abarbeitung von PILOT- Programmen

Zur Abarbeitung von PILOT- Programmen stehen die drei Prozeduren PILOTC, PILOTCL und PILOTCLG zur Verfügung.
Ergebnisse sind bei der Nutzung von PILOTC der PL1- Quelltext mit vorübersetzten Makros, von PILOTCL der ausführbare Lademodul und von PILOTclg die Ergebnisse des Programmes.
Die einzelnen Abarbeitungsschritte sind in der Abbildung 1 dargestellt.

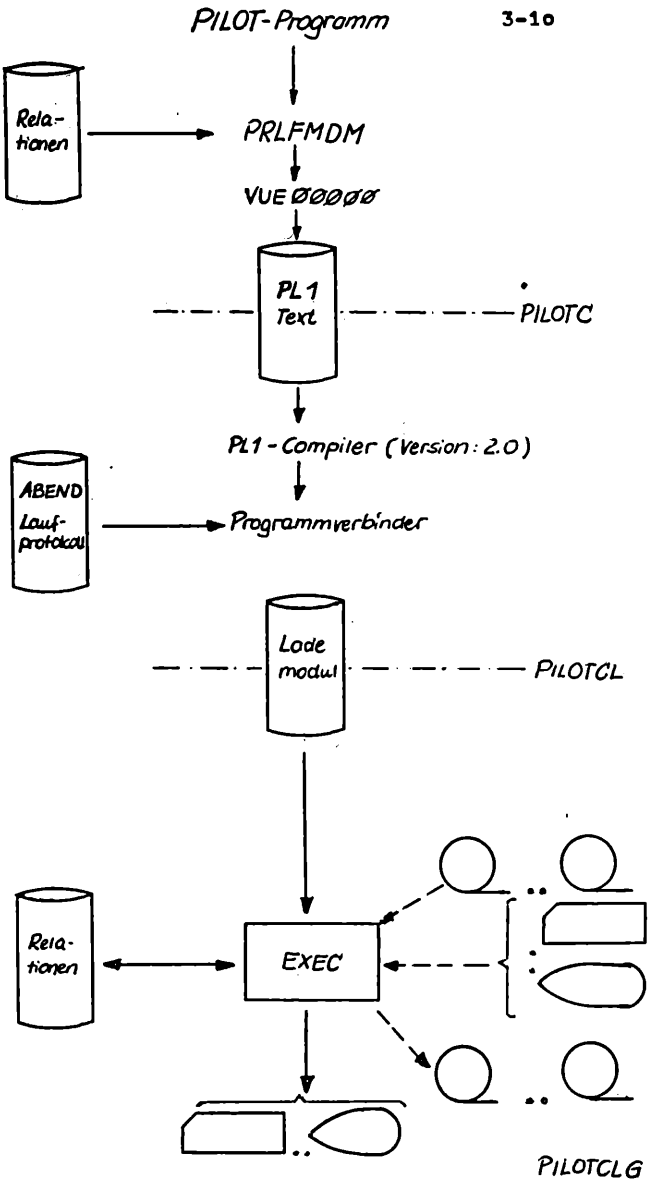

PILOT-Programm
Rela-
tionen
PRLFMDM
VUE ØØØØØ
PL 1
Text
PILOTC
PL1 - Compiler (Version: 2.0)
ABEND
Lauf-
protokoll
Programmverbinder
Lade
modul
PILOTCL
Rela-
tionen
EXEC
PILOTCLG

Abbildung 1

Literatur:

/1/ Falkenberg, E.: Architektur zukünftiger DBMS, Siemens Forschungs- und Entwicklungsbericht, 9 (1980)3

/2/ Astrahan, M.M.u.a.: Implementation of a structured english query language, Comm.of the ACM, 18 (1975) 10, S. 580-588

/3/ Pseudocode, VE KDV, HOP-01-024-1, 1981

/4/ Buchholz, B.: Definition und Implementierung einer nutzerorientierten Programmiersprache für das Datenbankbetriebssystem TOPAS und deren Anwendung für das Amt für Erfindungs- und Patentwesen der DDR, Dissertation, TU Dresden, 1984

/5/ Bormann, J., Lötzsch, J.: Definition und Realisierung von Fachsprachen mit DEPOT, Dissertation, TU Dresden, 1974

/6/ Das Datenbankbetriebssystem TOPAS, Anwenderdokumentation, VEB LfA Berlin, 1983

/7/ Anwenderbeschreibung PILOT VEB DVZ Frankfurt (Oder), 1984

Höllering, H.-H.; Schober, K.

Aufbau von Auskunftssystemen auf der Basis relationaler Datenbanken

Eine der Hauptaufgaben der Arbeit an den Rechenzentren des Bereiches des MHF besteht darin, die zentral gespeicherten Datenfonds der Hochschulverwaltung unmittelbar am Arbeitsplatz des zuständigen Sachbearbeiters verfügbar zu machen. Als hardwaremäßige Voraussetzungen dazu stehen relativ leistungsfähige Datenverarbeitungsanlagen der ESER-Reihe zur Verfügung (ES 1040).

Zur Lösung der oben beschriebenen Aufgabe gibt es verschiedene technische und technologische Möglichkeiten. Eine davon, die wir weiter verfolgt haben, ist der Anschluß von fernaufgestellten Bildschirmgeräten über die notwendige Hardware an unseren Zentralrechner ES 1040. Der Engpaß dieser Variante ist bekanntermaßen das Einrichten von Standleitungen durch die Deutsche Post, aber

- verfügt das ORZ über fernaufgestellte ES 7920 und ES 7925, und man kann hier testen (d. h. Sachbearbeiter kann auch im ORZ die BS-Technik nutzen) und

- ist der Engpaß in der Perspektive sicher abbaubar.

In der Perspektive werden auch Bürocomputer (BC)-Lösungen angestrebt, jedoch hängt dieses letztlich von der Verfügbarkeit entsprechender Hard- und Software ab.
Perspektivisch geht es um den Entwurf komplexer rechnergestützter Informationssysteme (kurz: RIS) unter Nutzung zentraler und dezentraler Rechentechnik am Arbeitsplatz, um die Vorteile beider Technologien auszunutzen und die Nachteile auszugleichen.
Neben der unmittelbaren Verfügbarkeit der Datenfonds für Datenverwaltungsprozesse des Sachbearbeiters wurde erstmalig angestrebt, einfache Anfragen im Dialog zu realisieren, also Datenverwaltungs- und Auskunftssysteme zu konzipieren.

Die Voraussetzungen dazu sind bei uns durch das Datenbankbetriebssystem (kurz: DBBS) "MIMER" gegeben, das vielen Nutzern bereits bekannt ist. Dieses System gestattet den Aufbau und die Verwaltung von Datenbanken, sowohl im Stapel als auch im Dialog, und ermöglicht die wenig aufwendige Programmierung einfacher Auskunftssysteme.

Lösungen auf der Basis anderer DBBS (DBS/R, Eigenentwicklungen) unter Nutzung des Teilhabersystems ESY/R wurden nach eigenen Testergebnissen aus Kapazitätsgründen verworfen, da unser Zentralrechner vorwiegend für die Forschung und Lehre verfügbar sein muß und erst in zweiter Linie für Verwaltungsaufgaben.

Als Projektierungsgegenstand wurden arbeitsökonomische Daten und Prozesse an einer Hochschule bzw. im Ministeriumsbereich ausgewählt. Ein erster Komplex wird durch die Stellenplan-Daten unserer Hochschule (MLU) gebildet. Darunter fallen sowohl die Planvorgaben und die Ist-Belegung des Stellenplanes als auch Informationen zu den Mitarbeitern der MLU. Dieser Datenfonds wurde aus dem vorhandenen Kaderprojekt und dem DBS/R-Stellenplanprojekt herausgezogen.

Ein zweiter Komplex enthält Informationen zum Arbeitskräfte- und

Lohnfondsplan bzw. -stand aller Hochschulen der DDR geordnet nach den Beschäftigtengruppen an den Einrichtungen (z. B. nach Professoren, Dozenten, Verwaltungspersonal usw.).
Beide Auskunfts- und Verwaltungssysteme (kurz: AVS) tragen gemeinsame Merkmale, die aus den Möglichkeiten und Grenzen des MIMER-Prozedurkonzeptes bzw. der MIMER-QL-Komponente resultieren (andere MIMER-Komponenten wurden in die Lösung nicht mit einbezogen):

- Datenfonds ist als MIMER-Datenbasis gespeichert (siehe Bild 1)
- Unterstützt werden alle Prozesse der Datenverwaltung und einfache Standardanfragen
- Single-user-System, d. h. kein Schutz vor konkurrierenden UPDATE (dieses läßt sich durch Übergang auf neue MIMER-Versionen bei Bedarf umgehen)
- Nutzung des AVS unter Steuerung des OS/ES-Teilnehmersystems TSO (TSO-Region: 180 K) oder als BTAM-Variante als normaler Job im OS/ES (Region: 190 K)
- MIMER-Prozedurkonzept unterstützt die Menütechnik
- Programmiersprache des AVS: MIMER-Prozeduranweisungen und MIMER-QL-Anweisungen der Relationensprache. Die so erzeugten Programme werden interpretativ abgearbeitet.
- Problemloser Wechsel von AVS zu MIMER-QL und umgekehrt möglich
- Ergebnisse der Verarbeitungen sind Zeilen mit Standardüberschriften und Nachrichten (keine Bilder)
- Kennwortschutz im AVS zu Nutzungsbeginn, Wahl der Verarbeitungsebene für den geübten und für den ungeübten Nutzer
- Hilfestellungen teilweise durch einbezogene Dokumentationsteile (z. B. Schlüsseltabellen, Codierregeln)

Unterschiede der beiden AVS:

Stellenplan-AVS

- Problemdatenfonds ca. 2 MByte, gegliedert in 5 Problemdatentabellen und 2 Sekundärindextabellen
- geschützte Prozeduren für privilegierte Nutzer

AK/LOHN-AVS:

- Problemdatenfonds ca. 500 KByte, gegliedert in 9 Tabellen
- Dokumentationsteil als Datenbank-Bestandteil

Beide AVS arbeiten in vielen Teilen analog. Das hier aufgeführte Beispiel (Bild 2) demonstriert die Arbeitsweise des AVS Arbeitskräfte/Lohnfonds. Es werden jeweils folgende Schritte ausgeführt:

1. Kennwortprüfung
2. Durchlaufgeschwindigkeitswahl
3. Wahl der Nutzungsart (Recherche/Verwaltung/Benutzungshilfestellungen)
4. Bestimmung der gewünschten Kennziffern und ihrer Ordnungsbegriffe

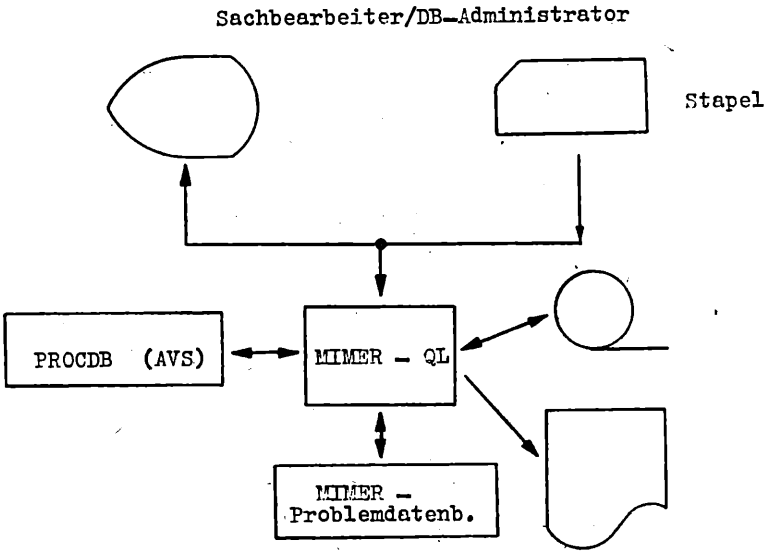


Bild 1 Systemstruktur der AVS

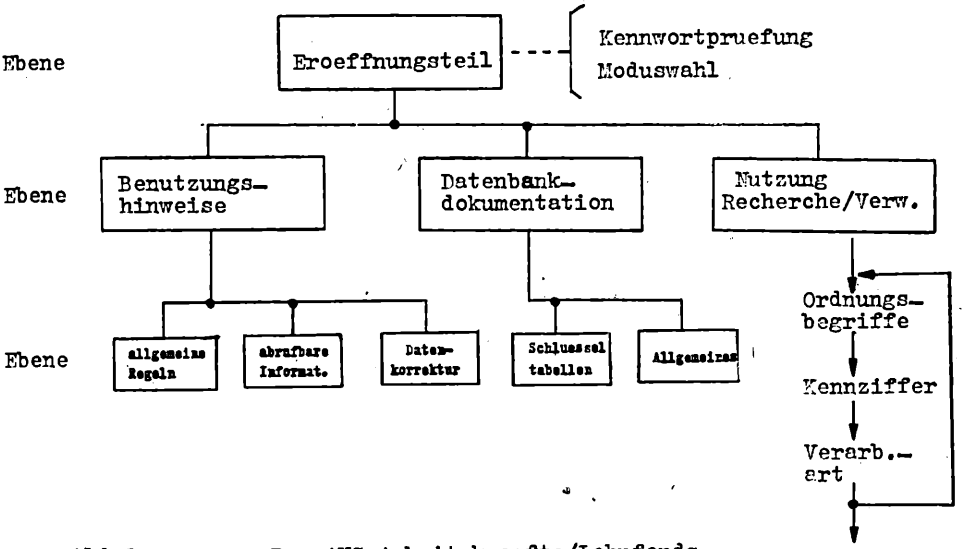


Bild 2 Das AVS Arbeitskraefte/Lohnfonds

Abbruch der aktuellen Verarbeitung und Verlassen der jeweiligen Verarbeitungsebene "nach oben" ist möglich.

Nach erstem Probebetrieb der AVS ist schon nachfolgender Nutzen bzw. sind schon nachfolgende Vorteile abzusehen:

- Schnelle operative Datenverwaltung durch Sofortverarbeitung im Dialog;
- Einfache Auskünfte sind operativ lösbar;
- Einfache und gleichsprachige Handhabung im Dialog wie im Stapelbetrieb;
- Einfache Programmierung der AVS, schneller Entwurf möglich
- Es ist eine unmittelbare Nutzung der AVS durch den Sachbearbeiter möglich;
- Fehlerbehandlung bei Fehleingaben und Fehlverarbeitung;
- Keine Konsistenzprobleme, da Änderungen unmittelbar in Datenbestand.

Als Nachteile haben sich erwiesen:

- Keine komfortable Verarbeitung im Sinne von Summierungen und Prozentberechnungen im AVS. Das liegt vor allem an den zur Realisierung verwendeten MIMER-Komponenten bzw. der vorliegenden MIMER-Version.
- Es sind nur Standardüberschriften bei Tabellenauswertungen möglich;
- Unerwünschte Systemaufschriften sind nur im Komplex unterdrückbar.

Zusammenfassung

Mit "MIMER" steht ein DBBS auf relatinnaler Basis zur Verfügung, das im Dialog und Stapelbetrieb gleichsprachige Anwendungen erlaubt. Die Datenbasis ist einfach erstellbar, schnell verwaltbar und es sind leichte, einfache (oder auch komplizierte) Recherchen möglich. Die Benutzung des DBBS ist leicht erlernbar und es ist relativ problemlos in der Handhabung. Der Aufbau und die Einführung der beschriebenen AVS zeigen die Nutzerfreundlichkeit für den Sachbearbeiter. Es wurde der Nachweis der Funktionsfähigkeit des DBBS bei relativ geringer Ressourcenbelastung erbracht, so daß das DBBS "MIMER" für umfangreiche und leistungsfähige Projekte anwendbar erscheint.

Dr. H.-H. Höllering

K. Schober

Krien, Reinhard:

Zur Hardwareunterstützung von relationalen Datenbanksystemen durch einen Assoziativprozessor

Die Effektivität eines Datenbanksystems wird wesentlich durch den schnellen und flexiblen Zugriff zu der Datenbasis bestimmt. Insbesondere erfordert die Verwirklichung des mengenorientierten Zugriffes in relationalen Datenbanksystemen eine Vielzahl von Externspeicherzugriffen, so daß die Antwortzeiten sehr schnell anwachsen können und zum Teil nicht mehr akzeptable Werte erreichen. Einen Ausweg aus dieser Situation stellt die international stark beachtete Entwicklung dar, Datenbanksysteme durch dedizierte Hardware zu unterstützen, bzw. sie in Datenbasismaschinen zu integrieren /1/, /2/. Bei dieser Hardwareunterstützung übernehmen assoziative Speicher- und Verarbeitungseinheiten wesentliche Aufgaben.
Der Assoziativprozessor AZP K 2064 ist eine Geräteeinheit für das Mikrorechnersystem MRS K 1600, der den inhaltsorientierten Zugriff auf ein Datenvolumen von 1 MByte gewährleistet /3/. Das Arbeitsprinzip des AZP gestattet es, solche Operationen der relationalen Algebra wie Selektion, Projektion und Join auf Hardwareebene auszuführen.
Es werden der Aufbau, das Funktionsprinzip sowie die Zusammenarbeit des AZP mit dem Betriebssystem MOOS 1600 dargestellt. Anhand eines Auskunftssystems auf der Basis des AZP werden einige Effekte diskutiert.

/1/ Manuel, T.: The Coming Surge in Data Base Systems
Electronics 57 (1984) 10 S. 131-142

/2/ Krien, R.: Datenbankmaschinen: Konzeption und Architektur
Rechentechnik/Datenverarb. 20 (1983) 5 S. 5-8

/3/ Krien, R.; Reichard, L.; Zweynert, M.: Assoziativprozessor AZP K 2064 zur Unterstützung von Such- und Speicheraufgaben im MRS K 1600
Neue Technik im Büro 28 (1984) 5 S. 148-149

Dr.-Ing. Reinhard Krien, VEB Robotron-Elektronik Dresden, Stammbetrieb, FG Rechentechnik, 8012 Dresden, PSF 330

Mollenhauer, Erhard

Entwicklungsstand eines VDBS im lokalen Rechnernetz

1. Einleitung

In einem solchen komplexen System, wie einem verteilten DBS, muß es eine Komponente geben, die eine zentrale Stelle einnimmt und auf der konzeptuellen Ebene angesiedelt ist. Sie koordiniert die Arbeiten aller anderen Komponenten des verteilten DBS und steuert die Realisierung der allgemeinen Funktionen der Nachrichtenübermittlung. Es handelt sich dabei um die VDBS-Komponente globales Datenbankbetriebssystem, die auf jedem Knoten des Rechnernetzes installiert ist. Diese Systemkomponente wird im weiteren als Dispatcher bezeichnet.

2. Struktur des Dispatchers

Dem Aufgabenspektrum eines verteilten DBS und den damit speziell vom Dispatcher umzusetzenden Aufgaben entsprechend, wird der Dispatcher modular aufgebaut (s. Bild 1).
Er setzt sich aus folgenden Moduln zusammen:

M1 dem Steuermodul DMON,
M2 dem Modul zur Analyse der Nutzeranforderung DSNUTZ,
M3 dem Modul zur Arbeit mit dem lokalen DBBS DSLDB,
M4 dem Modul zur Arbeit mit dem nichtlokalen DBBS DSNLDB,
M5 dem Modul zur Transformation Nutzersicht in die konzeptuelle Sicht DTRSF,
M6 dem Modul für den Aufbau und die Verwaltung des Katalogs DKAT,
M7 dem Modul für die Verteilung der Anforderungen DVERT,
M8 dem Modul für den Zugriffsschutz DSICH,
M9 dem Modul für die Transaktionsanalyse DTRAN,
M10 dem Modul für die Synchronisation DSYN,
M11 dem Modul für die Protokollierung DPROT,
M12 dem Modul für Fehlermitteilungen und -behandlung DFEHL,
M13 dem Modul für Recovery-Maßnahmen DREC.

Die einzelnen Moduln sind Einheiten, die eine oder mehrere Eingaben akzeptieren und durch Verarbeitung eine entsprechende Ausgabe produzieren. Mit den Moduln M1 bis M13 werden alle Aufgaben, die durch das globale DBBS zu erfüllen sind, bearbeitet.

Der modulare Aufbau läßt die Realisierung einer Überlagerungsstruktur zu und bietet somit die Möglichkeit einer speichereffektiven Implementierung. Aus Kapazitätsgründen ist es nicht möglich, alle Moduln in der Basisversion umzusetzen. Aus diesem Grund werden zunächst nur die zum Funktionsnachweis erforderlichen Moduln bearbeitet. Die im Bild 1 doppelt gerahmten Kästchen bedeuten, daß diese Moduln in der Basisversion implementiert werden, alle anderen sukzessive entsprechend der Realisierungskonzeption.

3. Systemkomponente Netzzugriffsprozeß

Der NZP dient als Interfaceprozeß zwischen dem auf den einzelnen Knoten befindlichen Dispatcher einerseits und dem Kommunikationssystem andererseits. Der NZP beinhaltet im Sinne des OSI-Referenzmodells die Anwendungsverwaltung und die Exekutive zur Kommunikationseinleitung und -durchführung. Entsprechend seiner Funktionen ist der NZP auf der Anwendungsschicht (Schicht 7) angesiedelt. Mit Hilfe des NZP wird es möglich, die für ein verteiltes DBS notwendigen Nachrichten über das Kommunikationssystem innerhalb des Netzes zwischen den Endsystemen auszutauschen. Zur Realisierung dieser Aufgabe bedient sich der NZP der von der Schicht 6 angebotenen Funktionen. Die Inanspruchnahme der Funktionen erfolgt mit Hilfe von Primitiven, in denen Parameter spezifiziert werden. Folgende Primitive stehen zur Verfügung:

- für die Sitzungseröffnung das P_CONNECT vom Typ request/response,
- für den Datentransport das P_DATA vom Typ request,
- für eine Veränderung von Bedingungen das P_PERFORM_NEGOTIATION vom Typ request/response,
- für eine vereinbarte Beendigung durch die Teilnehmer das P_RELEASE vom Typ request/response,
- für einen Abbruch durch den Teilnehmer das P_DISCONNECT vom Typ request,
- für den Abbruch durch den Sitzungsdienst P_ABORT vom Typ indication und
- für die Zusammenarbeit mit der Systemverwaltung das P_CONTROL vom Typ request/response.

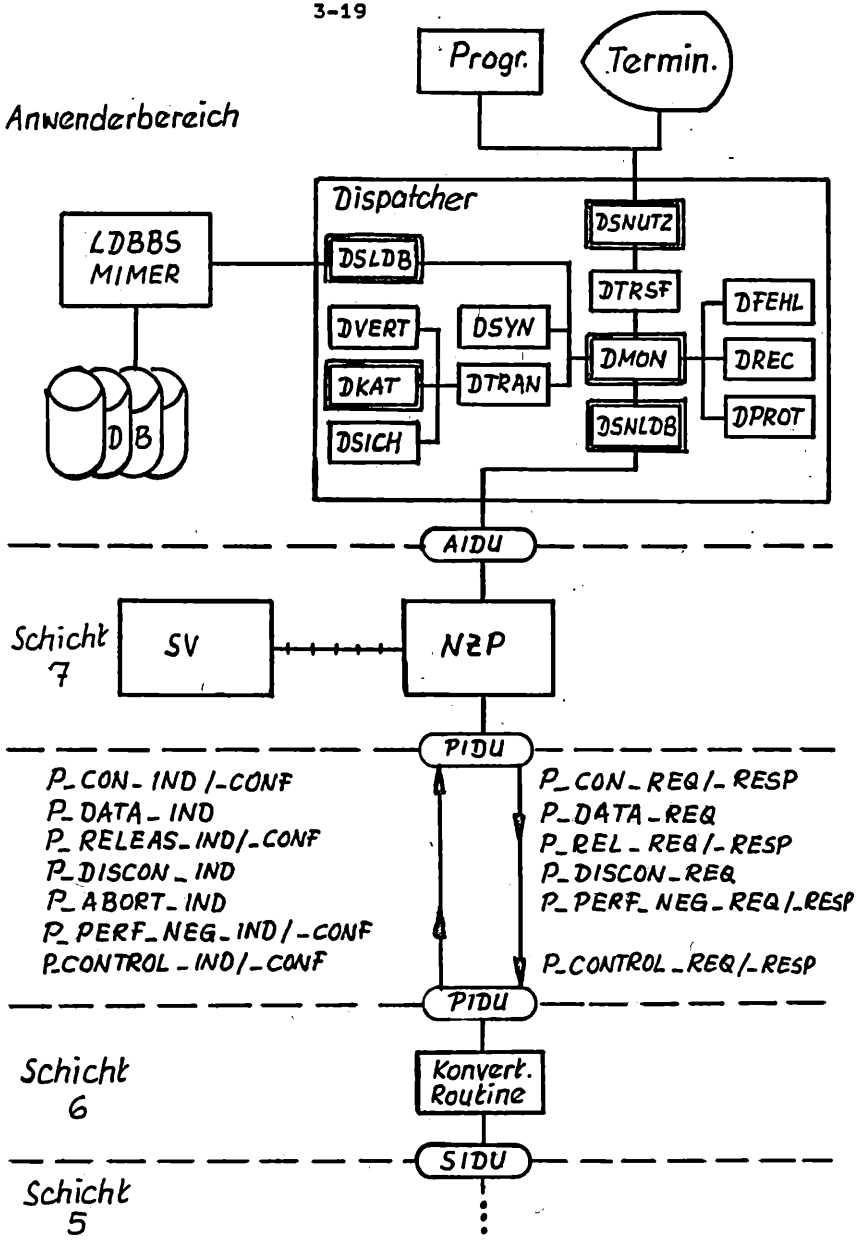

Progr.
Termin.
Anwenderbereich
Dispatcher
LDBBS
MIMER
DSLDB
DSNUTZ
DTRSF
DFEHL
DVERT
DSYN
DMON
DREC
DKAT
DTRAN
DSNLDB
DPROT
DSICH
D B
AIDU
Schicht
7
SV
NZP
PIDU
P_CON_IND/_CONF
P_DATA_IND
P_RELEAS_IND/_CONF
P_DISCON_IND
P_ABORT_IND
P_PERF_NEG_IND/_CONF
P_CONTROL_IND/_CONF
P_CON_REQ/_RESP
P_DATA_REQ
P_REL_REQ/_RESP
P_DISCON_REQ
P_PERF_NEG_REQ/_RESP
P_CONTROL_REQ/_RESP
PIDU
Schicht
6
Konvert.
Routine
SIDU
Schicht
5

Bild: 1

Der NZP muß Datenelemente für die Dienstdateneinheiten (SDU), Protokolldateneinheiten (PDU) und Interfacedateneinheiten (IDU) aufbauen, um

- Anforderungen über das LRN an entfernte Endsysteme senden und auswerten zu können,
- Anforderungen entfernter Endsysteme realisieren und entsprechende Antworten oder Mitteilungen an das anfordernde Endsystem übermitteln zu können.

Zur Erfüllung der Aufgaben der Anwendungsverwaltung muß der NZP eine Reihe von Übersichten in Form von Tabellen führen. Er führt

- eine Tabelle für bestehende Verbindungen als Initiator (VERBI),
- eine Tabelle für bestehende Verbindungen als Akzeptor (VERBA),
- eine Tabelle für gesendete Anforderungen (GA) und
- eine Tabelle für empfangene Anforderungen (EA).

Die Tabelle VERBI enthält den Initiatoridentifikator, die Akzeptorkennzeichnung mit Rechnernummer und Prozeßname des Dispatchers und den Akzeptoridentifikator.
Die Tabelle VERBA enthält den Akzeptoridentifikator, die Initiator-Kennzeichnung mit Rechnernummern und Prozeßname des Dispatchers und den Initiatoridentifikator.
Der Initiator- und Akzeptoridentifikator wird durch die Sitzungsschicht (Schicht 5) vergeben und dient der eindeutigen Zuordnung der PIDU'S zu den bestehenden Verbindungen. Der Identifikator ist dem Primitiv P_CONNECT_INDICATION und P_CONNECT_CONFORMATION zu entnehmen.
Die Tabelle GA besteht aus dem Nutzeridentifikator und der Kennzeichnung der Senke (je nach Art der Verbindung Akzeptor- bzw. Initiatoridentifikator) und die Tabelle EA aus dem Nutzeridentifikator und der Kennzeichnung der Quelle (je nach Art der Verbindung Initiator bzw. Akzeptoridentifikator).
Der Nutzeridentifikator dient der eindeutigen Zuordnung der Anforderungen, aber auch der Antworten bzw. Mitteilungen zu einem Nutzerprozeß und wird durch den Modul DSNUTZ vergeben.

Durch den NZP sind folgende Nachrichten und Operationen zu realisieren:

E AIDU vom Dispatcher (DSNLDB)
E PIDU vom Netz (Schicht 6)
V Auswerten einer AIDU
V Aufbauen einer AIDU
V Aufbauen einer PIDU
V Auswerten einer PIDU
V Führen der Tabellen
A PIDU an das Netz (Schicht 6)
A AIDU an den Dispatcher (DSNLDB)

4. Das Zusammenwirken der Komponenten im Netz

Für die Realisierung einer Forderung des Nutzerprozesses zur Arbeit mit den verteilt gespeicherten Daten müssen die beteiligten Komponenten (s. Bild 1) in folgender Art und Weise zusammen arbeiten.

Nach dem Start des Dispatchers muß dieser sich beim Netz anmelden. Der Nutzerprozeß übergibt an den Dispatcher (DSNUTZ) eine Anforderung. Die Anforderung wird durch den Nutzer- und Nachrichtenidentifikator charakterisiert und in die WS eingetragen. Danach wird die Transaktionsanalyse (DTRAN) durchgeführt und festgestellt, ob der Nutzerprozeß zugriffsberechtigt ist und wo die Daten sich befinden. Sind die Daten lokal gespeichert, dann wird die Transaktion vom Dispatcher (DSLDB) an das lokale DBBS übergeben und die zu erwartende Antwort bzw. Mitteilung wird durch den Dispatcher (DSNUTZ) aufgrund der geführten Tabellen und bestehenden Identifikatoren an den Nutzerprozeß übergeben.

Wird festgestellt, daß die Daten nicht lokal gespeichert sind, wird die Transaktion durch den Dispatcher (DSNLDB) in Form einer A-Interfacedateneinheit an den NZP übergeben. Der NZP baut zunächst eine PIDU mit dem P_CON_REQ-Primitiv auf, um eine Verbindung mit dem gewünschten Endsystem herzustellen. Kommt die Bestätigung des Verbindungsaufbaues (P_CON_CONF-Primitiv), so kann der NZP eine PIDU mit dem P_DATA_REQ-Primitiv, in dem die Anforderung enthalten ist, an das Netz (Schicht 6) übergeben und somit den Zugriff zu den entfernt

gespeicherten Daten ermöglichen. Als Antwort erhält der NZP eine PIDU mit P_DATA_IND, die er auswertet und in Form einer AIDU dem Dispatcher (DSNLDB) übergibt.
In umgekehrter Richtung empfängt der NZP eine PIDU mit P_CON_IND, womit mit ihm eine Verbindung gefordert wird. Als Antwort sendet er eine PIDU mit P_CON_RESP mit dem Einverständnis oder Ablehnung an den initiierenden Knoten. Bei Einverständnis wird er eine PIDU mit P_DATA_IND mit der entsprechenden Forderung zum Zugriff zu den Daten auf seinem Knoten empfangen. Diese Anforderung wird an den Dispatcher (DSNLDB) in einer AIDU übergeben und von ihm (DSLDB) an das LDBBS weitergegeben. Die Antwort bzw. Mitteilung geht den entgegengesetzten Weg an den NZP und von diesem in einer PIDU mit P_DATA_REQ an den initiierenden Knoten zurück.
Wird bei der Verbindungsaufnahme in der Antwort im P_CON_CONF im Diagnoseparameter mitgeteilt, daß der gewünschte Prozeß (Dispatcher) nicht aktiv (nicht bekannt) ist, so wird vom NZP durch Aufbau und Absenden einer PIDU mit P_CONTROL_REQ an die Systemverwaltung, diese aufgefordert, den Prozeß auf dem gewünschten Endsystem zu starten. Erhält der NZP durch eine PIDU mit P_CONTROL_CONF eine positive Antwort, kann er erneut ein P_CON_REQ zur Verbindungsaufnahme absenden.
Soll die Verbindung abgebaut werden, so wird eine PIDU mit P_REL_REQ abgesetzt und die Bestätigung durch P_REL_CONF abgewartet, bevor die Verbindung aus den Tabellen ausgetragen wird. Nachdem alle Verbindungen abgebaut sind und der Dispatcher seine Arbeit beenden will, muß er sich vom Netz abmelden. Es kann während der Arbeit auch ein Verbindungsabbruch durch P_DISCON_IND oder P_ABORT_IND auftreten.

Die Konvertierung aufgrund des heterogenen Netzes erfolgt in der Schicht 6. Zu diesem Zweck werden im P_CON_REQ bestimmte Parameter spezifiziert. Ergeben sich während des Bestehens der Verbindung Änderungen, so können diese mit dem P_PERFORM_NEGOTIATION_REQ mitgeteilt werden.

Damit die NZP auf den einzelnen Knoten sich verstehen, was mit den ankommenden Nachrichten zu tun ist, werden A-Protokolle ausgetauscht, die z. Z. nur den Nutzer- und Nachrich-

tenidentifikator enthalten. Die Nachrichtenlänge und der Initiator- und Akzeptoridentifikator können der P-Interfacesteuerinformation (PICI), die die ersten 9 Byte der PIDU belegt, entnommen werden.

Autor:

Dipl.-Ing. Mollenhauer, Erhard

Ingenieurhochschule Dresden
Sektion Informationsverarbeitung
8019 Dresden
Hans-Grundig-Str. 25

Pälecke, Bernd

Recovery für Datenbasen, die im Dialog aktualisiert werden

Die z.Zt gebräuchlichen Systeme zur Aktualisierung großer Datenbasen enthalten meist nur spärliche Vorkehrungen für den Fehlerfall. Moderne Methoden zur Manipulation von Datenbasen (z.B. Dialogsysteme) kommen ohne Maßnahmen zur Fehlerbehandlung nicht aus.

Ausgehend von einem Fehlermodell werden im Vortrag die einzelnen Recovery-Fälle für eine Datenbasis aufgezeigt und behandelt.

Wie mit Hilfe der Auflösung von Datenmanipulationen in Transaktionen und der Bereitstellung entsprechender Tools Recovery realisiert werden kann, wird am Beispiel des Systems BASTARD gezeigt.

Bernd Pälecke
Karl-Marx-Universität Leipzig, ORZ
7o1 Leipzig
Karl-Marx-Platz

Porkert, Kurt

Entwerfen der Grobstruktur der Datenbasis zur Leitung einer Wirtschaftseinheit

Für ein Industriekombinat und seine Betriebe ist zu konzipieren, wie die Arbeitsteilung bei der maschinellen Datenverarbeitung zur Unterstützung der Leitung langfristig gestaltet werden soll. Das verlangt unter anderem, die Struktur der Datenbasis im Zusammenhang mit der Projektgliederung und der Aufteilung der technischen Mittel zu entwerfen.

Eine Datenbasis für die Leitung stellt ein Modell des Leitungsobjekts und seiner Beziehungen zur Umgebung dar, wie es für das Lösen der Leitungsaufgaben relevant ist. Sie repräsentiert wesentliche Eigenschaften einzelner Objekte durch Datensätze, -felder und -gruppen. Klassen differenziert aggregierter Objekte und deren Beziehungen werden durch Satzmengen und Satzmengenrelationen widergespiegelt. Die Gesamtheit der Beziehungen zwischen Satzmengen einer Datenbasis soll als Grobstruktur der Datenbasis bezeichnet werden. Durch einzelne Leitungsaufgaben sind externe Datenbasen und durch die Gesamtheit berücksichtigter Aufgaben die konzeptionelle Datenbasis begründet. Die interne Datenbasis verdeutlicht eine Datenstruktur, die ablauf- und speicherorganisatorischen Anforderungen entspricht.

In der Literatur wird vorgeschlagen, die Datenbasis nach Prädikaten durch horizontale und vertikale Partitionierung von Datenrelationen aufzuteilen. Dabei entstehen entweder Teilmengen für Objekte, die hinsichtlich ihrer Abbildungserfordernisse einer gemeinsamen Grundgesamtheit angehören, oder Teilmengen für Objekte mit unterschiedlichen Attributen beziehungsweise für differenzierte Attribute zu Objekten. Das satzweise Aufteilen nach einem eindeutigen Schlüssel oder nach Zugriffsstatistiken wird als wenig praktikabel bewertet.

Bekannte Methoden und Verfahren für das Entwerfen der Struktur einer Datenbasis gehen

- von einer semantischen Analyse der Beziehungen zwischen Attributen aus, die zum Bestimmen von Satzmengen führt, oder
- von der Synthese der Satzmengen und ihrer Beziehungen aus, an die sich Attributszuweisungen zu Satzmengen anschließen.

Das Ergebnis der semantischen Analyse der Beziehungen zwischen Attributen anhand der Attributsnamen ist jedoch nur insofern variabel, wie unterschiedlich benannte Schlüsselattribute verschiedene Objektklassen vertreten. Daher führen sie in der Regel zum Bilden von Satzmengen für Objektklassen, zwischen denen keine oder solche Beziehungen bestehen, daß Attribute von Objekten verschiedener Klassen auch als ein Satz eines aggregierten Objekts angesehen werden können. Verschiedene Satzmengen für Objekte, die hinsichtlich der abzubildenden Attribute einer gemeinsamen Grundgesamtheit angehören, ergeben sich daraus nicht.

Solche Satzmengen existieren aber vielfach für die Leitung einer Wirtschaftseinheit. Sie resultieren daraus, daß zwischen den abgebildeten Prozessen in der Wirtschaftseinheit hierarische Beziehungen bestehen und in verschiedenen Prozessen Objekte einer gemeinsamen Grundgesamtheit enthalten sind. Beispiele sind Dateien für "Arbeitsgegenstände eines Betriebes", "Arbeitsgegenstände eines Produktionsbereiches" oder "Arbeitsgegenstände eines Produktionsabschnitts" beziehungsweise Dateien für "Kennziffern sämtlicher Kostenstellen" oder "Kennziffern der Kostenstellen der Vorfertigung". Aufgrund der Arbeitsteilung zwischen den Leitungsorganen, unterscheiden sich externe Datenbasen zu Leitungsaufgaben, die auf unterschiedliche Prozesse, aber partiell auf dieselben Objekte bezogen sind, oft erheblich durch die Attribute. Das begründet die Nützlichkeit, nach Prozessen differenzierte Satzmengen für ähnliche Objekte sichtbar

zu machen, und ist bei der Synthese der Modelle (Schemata) der konzeptionellen und der internen Datenbasis zu berücksichtigen.
Dies kann durch eine Synthese erfolgen, die von benannten Satzmengen ausgeht. Sie kann sich auf Entity-Relationship-Methoden stützen. Aus der Literatur sind allerdings ausreichende Hinweise nicht bekannt, wie mit Hilfe eines geeigneten Modells der konzeptionellen Datenbasis eine Struktur einer verteilten internen Datenbasis entworfen werden kann, die einen hohen Nutzeffekt der Datenverarbeitung gewährleistet. Nachfolgend wird ein Ablauf des Entwerfens vorgestellt, wie es zum Ermitteln der Grobstruktur der Datenbasis für die Leitung der Produktionsdurchführung in einem Kombinat der metallverarbeitenden Industrie exerxiert wurde.

Der Ablauf der Synthese von Modellen zu Datenbasen ist durch die vorgesehene Verwendung der Modelle determiniert. Das Modell der konzeptionellen Datenbasis sollte als Grundlage für die Synthese des Modells der internen Datenbasis dienen und zugleich die Analyse des Niveaus der Datenverarbeitung, der Gemeinsamkeiten und Unterschiede des Leitungsobjekts und der Leitungsaufgaben zu anderen Wirtschaftseinheiten, zukünftiger Strukturen des Leitungsobjekts, der Verträglichkeit der Anwendungsbedingungen vorgefertigter Datenverarbeitungslösungen mit vorliegenden Voraussetzungen sowie die Synthese von Leitungsaufgaben und ihre Zuordnung zu Leitungsstruktureinheiten unterstützen.
Diesen Aufgaben wird das Modell nur gerecht, wenn es die für die Leitung relevanten Prozesse und Klassen der Prozeßelemente sowie die Beziehungen zwischen den Klassen sichtbar macht. Dies ist durch ein Vorgehen zu erreichen, das im Unterschied zu anderen Methoden auch Satzmengen zu verschiedenen Klassen von Objekten einer Grundgesamtheit mit den Relationen zwischen ihnen ausweist. Es verläuft in den Schritten

- Aufstellen der Modelle externer Datenbasen,
- Zusammenfügen dieser Modelle und
- Prüfen und Erweitern der Vollständigkeit des Modells hinsichtlich zu berücksichtigender Leitungsaufgaben.

Voraussetzung dafür ist, daß beim Aufstellen der Modelle externer Datenbasen bereits Satzmengen bestimmt und exakt definiert werden. Folgt das Entwerfen der Datenbasis der Dekomposition der Datenverarbeitungsaufgaben für die Leitung nach dem Top-down-Prinzip wird zuerst die vorläufige Grobstruktur einer externen Datenbasis sowie danach ihre Fein- und endgültige Grobstruktur bestimmt. Die Grobstruktur entsteht dadurch, daß die Leitungsaufgabe durch Satzmengen für zugeordnete Objektklassen und deren Beziehungen beschrieben wird. In der Literatur heißen diese Modelle auch "lokale" oder "funktionelle Entity-Modelle".

Vielen Projektanten fällt es leichter, mit der Zusammenstellung der Attribute einer externen Datenbasis zu beginnen und sie danach in Satzmengen zu ordnen. Das Ermitteln der Feinstruktur schließt das Prüfen der Attribute auf Zerlegbarkeit ein. Für jedes Attribut ist festzustellen, ob es einer der bereits definierten Objektklassen direkt zugeordnet werden kann oder ob es ein primäres Attribut einer anderen Klasse aggregierter Objekte darstellt.

Beim Zusammenfügen der Modelle externer Datenbasen zum Modell der konzeptionellen Datenbasis wird zu jeder Objektklasse, die für Leitungsaufgaben relevant ist, eine Satzmenge definiert. Beziehungen zwischen Satzmengen werden entsprechend den Objektbeziehungen deklariert. Ein Modell der konzeptionellen Datenbasis entsteht durch das

- Prüfen der in den externen Modellen enthaltenen Bezeichnungen für Satzmengen und Attribute auf Homonymität und Synonymität,
- Beseitigen homonymer und synonymer Bezeichnungen,
- Übertragen der unterschiedlichen Bezeichnungen für Satzmengen, die den Objektklassenbezug sichtbar machen, aus den Modellen der externen Datenbasen,
- Zusammenfassen der in den Modellen zu einer Satzmenge enthaltenen Bezeichnungen für Attribute und Übertragen der unterschiedlichen Bezeichnungen zu jeder Satzmenge,

- Ableiten und Übertragen einer Bezeichnung für die Beziehung zwischen zwei Satzmengen, zwischen denen in den Modellen der externen Datenbasis übereinstimmende oder unterschiedliche Beziehungen gekennzeichnet sind sowie durch das
- Kennzeichnen der Beziehungen zwischen Satzmengen, die als Mengen einer gemeinsamen Grundgesamtheit anzusehen sind.

Die Aussagekraft des Modells für verschiedene Analysen erhöht sich mit zunehmender Vollständigkeit hinsichtlich der gegenwärtig und zukünftig zu lösenden Leitungsaufgaben. Damit wird zugleich dem nachträglichen Änderungsaufwand für die Datenbasisstruktur entgegengewirkt, da die Mehrzahl der Änderungen aus unvollkommenen Ergebnissen eines Bottom-up-Vorgehens oder aus der Entwicklung des Leitungsobjekts und der Leitungsorganisation herrühren. Das durch Zusammenfügen vorliegender Modelle externer Datenbasen entstandene Modell der konzeptionellen Datenbasis wird daher aufgrund von Vergleichen mit analogen Modellen anderer Wirtschaftseinheiten, mit Modellen aus der Literatur, mit anderen Modellen des Leitungsobjekts und mit Modellen zur Entwicklung des Leitungsobjekts vervollkommnet. Die Vervollkommnung ist mit Entscheidungen über die Vorgabe von Leitungsaufgaben verknüpft.
Es ist eine interne Datenbasis anzustreben, die eine hohe Reaktionsfähigkeit der Leitungsorgane bei geringem Aufwand ermöglicht. Eine solche Datenbasis kann durch Nachnutzen effektiver Lösungen oder durch Variantenvergleich gefunden werden.
Modelle interner Datenbasen werden vorwiegend in Verbindung mit Programmlösungen nachgenutzt. Die Übernahme eines Modells setzt voraus, daß der Nutzeffekt der enstprechenden Datenbasis nachgewiesen wurde, es sich auf ähnliche leitungsorganisatorische Voraussetzungen bezieht und seine Umsetzung praktikabel ist.
Anderenfalls wird das Aufstellen und Vergleichen von Varianten unumgänglich. Varianten der internen Datenbasis ergeben sich grundsätzlich dadurch, daß im Modell der konzeptionellen Daten-

basis enthaltene Festlegungen übernommen oder das Auftreten bestimmter Daten, von Daten, die aus anderen ableitbar sind, und von Relationen sowie die Zuordnung von Satzmengen zu Objektklassen gegenüber der konzeptionellen Datenbasis geändert wird. Der Syntheseaufwand läßt sich einschränken, wenn vorrangig solche Varianten aufgestellt werden, deren Auswahl überdurchschnittlich wahrscheinlich ist. Dem wird mit einer Satzmengendefinition entsprochen, die neben der Übernahme vor allem auf der Elimination von Satzmengen des Modells der konzeptionellen Datenbasis beruht, die nachfolgend klassifiziert wird.

Arten der Elimination von Satzmengen

- die vollständig redundante Teilmengen einer anderen Satzmenge darstellen,
- die zu Satzmengen vollständig redundant sind, die Teilmengen der Objekte einer Grundgesamtheit enthalten,
- deren Daten vollständig durch die Verknüpfung von Daten zu anderen Objekten ableitbar sind,
- die einen Teil der Objekte vollständig redundant zu anderen Satzmengen abbilden, wobei Satzmengen für disjunkte Objekte
  - abgetrennt oder
  - nicht abgetrennt werden,
- die sämtliche Objekte teilweise redundant zu anderen Satzmengen abbilden, wobei Satzmengen für disjunkte Attribute
  - abgetrennt oder
  - nicht abgetrennt werden,
- die Teilmengen zu disjunkten Objekten einer Grundgesamtheit darstellen beziehungsweise
- deren Daten Satzmengen zu aggregierten Objekten desselben Prozesses zugeordnet werden.

Beispiele sind Fertigungsauftragsdateien für einen Produktionsbereich und für diesen untergeordnete Abteilungen, die differenziert redundant sein können. Die Daten der Dateien "Arbeitsgänge" und "Arbeitsorte" können auch der Datei "Fertigungsaufträge" zugeordnet sein. Diese Variationen werden durch das Hin-

zufügen von Satzmengen für ausgewählte Attribute zu Objekten und durch Elimination von Relationen zwischen Satzmengen ergänzt, die im Modell der konzeptionellen Datenbasis enthalten sind.

Um die Anzahl aufzustellender und zu bewertender alternativer Varianten und damit den Entwurfsaufwand gering zu halten, folgt die Auswahl der Aufstellung von Varianten in einer Reihenfolge, bei der durch Eliminationsentscheidungen, die Anzahl nachfolgender Entscheidungen eingeschränkt wird.

```
Variantendefinition und -auswahl
Beginn
  Wiederhole für alle Variationsarten
    Wiederhole für alle vertikalen Leitungsbeziehungen
      Wiederhole für alle horizontalen Leitungsbeziehungen
        Bestimme Satzmengen, die eliminiert oder hinzuge-
        fügt werden können!
        Bilde Varianten der Elimination von Satzmengen und
        Relationen sowie des Hinzufügens von Satzmengen!
        Ordne die alternativen Varianten in einer Rangfolge
        nach fallendem Nutzeffekt!
    Kombiniere Varianten zur Grobstruktur des Modells der
    internen Datenbasis!
Ende
```

Alternative Varianten zum Übertragen von Satzmengen aus dem Modell der konzeptionellen Datenbasis ergeben sich durch ihre Elimination mit oder ohne Hinzufügen anderer Satzmengen beziehungsweise durch Elimination anderer Satzmengen derselben Grundgesamtheit mit oder ohne Hinzufügen weiterer Satzmengen. Varianten, die in einem Vergleich als weniger effektiv eingestuft wurden, sollten bei nachfolgenden Variantendefinitionen vernachlässigt werden.

Es werden nur solche Abweichungen vom Modell der konzeptionellen Datenbasis bestätigt, die zur Gewährleistung nachfolgend aufgeführter zeitlicher, räumlicher und anderer Erfordernisse der Aufgabenlösung oder aus personellen, technischen beziehungsweise ökonomischen Gründen erforderlich werden oder die nachweisbare Aufwandssenkungen bewirken.

Restriktionen für die Variantenaufstellung und -auswahl

Erfordernisse der Leitungsaufgaben:

- durchschnittliche Anzahl aus einer Satzmenge benötigter Sätze
- primäre und sekundäre Sortierung der auszugebenden Sätze
- Häufigkeit des Lösen einer Leitungsaufgabe je Zeiteinheit
- zeitliche Verteilung der Aufgabenlösung je Zeiteinheit
- maximal zulässige Reaktionszeit beim Lösen der Leitungsaufgabe
- geforderte Aktualität zu verarbeitender Daten
- Zuordnung von Leitungsaufgaben zu Leitungsorganen
- Geheimhaltung von Daten
- räumliche Verteilung der maschinellen Verarbeitung und Datenausgabe an die Leitungsstruktureinheiten

verwendbare technische Mittel:

- Art und Anzahl der Geräte
- Konfiguration
- Betriebssysteme
- Datenträgerarten oder -anzahl

Leistungsfähigkeit und Kostensätze der technischen Mittel:

- Operations-, Verarbeitungs- oder E/A-Geschwindigkeiten
- Zugriffs- und Reaktionszeiten
- Speicherkapazitäten
- Fehlerrate und Verfügbarkeit
- Kostensätze je Betriebszeiteinheit

Konventionen verwendeter Betriebssysteme:

- zulässige Anzahl an Relationen, in denen eine Satzmenge enthalten sein kann
- zulässige Anzahl von Hierarchiestufen
- zulässige Anzahl an Datei- oder Segmenttypen
- zulässige Anzahl an Elementen in Dateien oder Segmenten

personenbezogene Voraussetzungen:

- vorhandene Zeitfonds- und Qualifikationsstruktur der Arbeitskräfte
- Bedien- und Wartungsaufwand
- Bedien- und Wartungsleistung
- Qualifikationsanforderungen
- Forderungen bezüglich der Arbeitsbedingungen
- Arbeitskräftelimit

ökonomische Vorgaben:

- Kostenlimit

Die Satzmengen der internen Datenbasis werden zusammen mit den Zugriffspfaden und den Datenflüssen festgelegt, was Entscheidungen über die Abgrenzung der längerfristig als Stammdaten zu speichernden Satzmengen, der aus diesen Satzmengen durch Verarbeitung abgeleiteten und temporär als Arbeitsdateien zu gestaltenden Satzmengen sowie der Satzmengen für die Datenerfassung einschließt. Dem entspricht, Varianten der Datenbasisstruktur mit Prozeßablaufmodellen zu verknüpfen. Veränderte Modelle der Grobstruktur der Datenbasis bedingen stets neue Prozeßabläufe. Einem Strukturmodell können jedoch verschiedene Prozeßablaufmodelle zugeordnet werden. Vor allem in Abhängigkeit von der Anzahl sowie von der zeitlichen und räumlichen Verteilung der Prozesse für die Speicherung und Auswertung, von den Zugriffs- und Verarbeitungszeiten sowie von den Kostensätzen je Zeiteinheit verändert sich der Nutzeffekt einer bestimmten Datenbasisstruktur. Daher bezieht sich der Vorzug einer bestimmten Variante gegenüber einer anderen auf unterstellte Prozeßabläufe.
Für die Auswahl einer Variante sind in vielen Fällen Aussagen über erwartete Aufwände entscheidend. Voraussetzung dafür ist, daß die alternativen Varianten annähernd gleiche Bedingungen für die Reaktionsfähigkeit der Leitungsorgane schaffen. Auswirkungen von Varianten lassen sich oft auf die

- Anzahl und Kompliziertheit der Arbeitsgänge und Programme,
- Zeiten für Datenein- und-ausgaben zu einem Arbeitsgang,
- Zeiten für Datenein- und-ausgaben über alle Arbeitsgänge je Zeiteinheit,
- Speicherplatzbedarfsstruktur und auf die
- Kosten beim Datenträgereinsatz

feststellen. Eine Elimination von Satzmengen des Modells der konzeptionellen Datenbasis ist vorzuschlagen, wenn infolge verringerter Datenredundanz erzielbare Einsparungen an Speicherungsaufwand höher zu werten sind als unter Umständen infolge anwachsender Ballastanteile oder steigender Anzahl auszuwertender Satzmengen auftretende zusätzliche Auswertungsaufwände, sofern die Ausprägung anderer Nutzeffektsmerkmale nicht untersucht wird.

Das Hinzufügen von Satzmengen gegenüber dem Modell der konzeptionellen Datenbasis ist gerechtfertigt, wenn sich dadurch der Auswertungsaufwand stärker verringern läßt, als der Speicherungsaufwand gleichzeitig steigt.
Redundanzarme Datenbasen besitzen in der Einheit mit einer zentralisierten maschinellen Datenverarbeitung überwiegend mehr Vorteile als in Verbindung mit Formen verteilter Verarbeitung. Demgegenüber wirkt eine teilweise oder eine vollständig verteilte Verarbeitung vielfach mit Datenbasen vorteilhaft zusammen, in denen mehrere Satzmengen zu Objekten einer Grundgesamtheit vorliegen. Daher liefern die Klassen von Elementen der für die Leitungsorgane relevanten Prozesse wesentliche Prädikate für Entscheidungen über die räumliche Verteilung der Speicherung und Verarbeitung von Daten.
Als Entscheidungshilfe dienen Datenbasisdiagramme, die Relationen zwischen Satzmengen für Objekte eines Prozesses und zwischen Satzmengen für Objekte mehrerer Prozesse sichtbar machen. Ein Datenwörterbuch, das gegenwärtige und zukünftige Strukturen des Leitungsobjekts sowie ihren Zusammenhang zu Leitungsaufgaben und -organen verdeutlicht, stellt ein weiteres Werkzeug dar.

Für die Leitung der Produktionsdurchführung im untersuchten Industriekombinat entstehen Modelle der konzeptionellen und der internen Datenbasis mit Satzmengen, die nach vier Leitungsebenen getrennt sind. Das Modell der internen Datenbasis weist einige Satzmengen aus, beispielsweise zu Arbeitsgegenständen, Arbeitsplänen, Stücklisten, Arbeitsorten, Arbeitskräften oder Grundmitteln, die durch mehrere Leitungsorgane auszuwerten sind. Andere Satzmengen, zum Beispiel für Fertigungsaufträge, Zeitfonds, Kosten oder Plankennziffern, werden vorwiegend durch einzelne Leitungsorgane genutzt. Es bietet sich an, diese Datenbasis in Form einer teilweise zentralisierten und teilweise räumlich verteilten Verarbeitung zu verwenden.

Dr. Kurt Porkert
Friedrich-Schiller-Universität
Sektion Wirtschaftswissenschaften
6900 Jena
Schillerstr. 8

Prescher, Gerold

Datenbankentwurf mit dem Entity- Relationship- Modell

1. Einleitung

Die Effektivität von Datenbanken ist heute vor allem die Effektivität ihres Entwurfes. Am Problem des Datenbankentwurfes wird weltweit gearbeitet (siehe z. B. / 1 / / 2 / / 3 /).
Das relationale Datenmodell und die Theorie der funktionalen Abhängigkeiten (Functional Dependency, Abkürzung: FD) und der mehrdeutigen Abhängigkeiten (Multivalued Dependency, Abkürzung: MVD) werden im Datenbankentwurf angewendet.
Bisherige Ansätze für den Datenbankentwurf (z. B. / 1 / / 4 /) setzen voraus, daß eine Menge funktionaler und/oder mehrdeutiger Abhängigkeiten zur Verfügung steht. Arbeitsgegenstand des Entwurfsprozesses ist dann die Manipulation der Abhängigkeiten mit dem Ziel des Aufbaus einer Relationenstruktur, die der Coddschen Normalformenlehre gehorcht. Geht man davon aus, daß der Aufbau einer Datenbank in seiner ersten Phase die nichtprozedurale Darstellung eines erhobenen Ausschnittes der objektiven Realität unter ständiger Beachtung des Zweckes der Datenbank ist, so ergibt sich, daß der Datenbankentwurf nicht allein Aufgabe des Informatikers sein kann. Der Entwurf der Datenbank muß vielmehr aus dem Sachgebiet heraus erfolgen, dessen Bearbeiter auch den Zweck der Datenbank formuliert.
In dieser Phase der Modellierung kann schwerlich schon die Ableitung der vom Informatiker benötigten Abhängigkeiten durch den Bearbeiter des Sachgebietes erwartet werden. Es geht hierbei um die erste Erhebung von Interessenobjekten und des zwischen ihnen bestehenden Geflechtes von Beziehungen. Unterschätzung der Bedeutung dieser Phase, fehlende Hilfsmittel und mangelhafte Zusammenarbeit Sachgebietsspezialist - EDV- Fachmann haben z. T. zu einem Anwendungsstau auf dem Gebiet der Datenbanktechnologie bzw. zu uneffektiven Datenbankentwürfen geführt. Deshalb sind beim Datenbankentwurf Modelle heranzuziehen, die die Formalisierung der erhobenen zweckrelevanten Semantik

des Sachgebietes mit dem Ziel der Gewinnung der Datenbankstruktur unterstützen. Das Entity- Relationship- Modell / 5 / (im folgenden wird dafür auch die Abkürzung ERM verwendet) wurde auf seine diesbezügliche Eignung untersucht (s. a. / 6 /).

2. Das Entity- Relationship- Modell als Darstellungsmittel

Das ERM eignet sich für die Etappe der Fixierung von Modellvorstellungen in einem Sachgebiet (Erhebungsphase). In seiner Darstellungskraft und semantischen Reichhaltigkeit wurde es bereits vielseitig erprobt (siehe z. B. / 7 /). Das Entity- Relationship- Modell beruht auf der Darstellung von Objekten (Entities) und der zwischen ihnen bestehenden Beziehungen (Relationships), die als 1 : 1, 1 : N und M : N- Beziehungen typisiert sind. Zwischen je zwei der als Rechtecke dargestellten Entities (s. a. Abb. 1) kann mehr als eine Relationship (durch Rhomben gekennzeichnet) bestehen; in diesem Fall spricht man von Rollen. Entities und Relationships werden benannt und können Attribute (durch Kreise versinnbildlicht) besitzen. In Abb. 1 wurden aus Übersichtlichkeitsgründen keine Attribute aufgenommen.
Vorzüge des ERM sind Übersichtlichkeit und Klarheit der semantischen Darstellungsmöglichkeiten. Der sachliche Hintergrund in Abb. 1 ist das vereinfachte Modell einer Abteilungsorganisation.

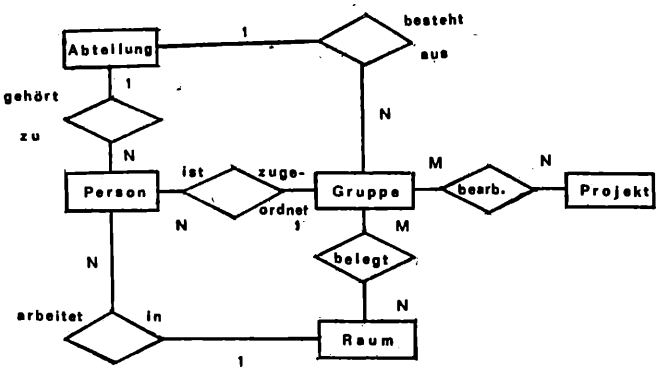


Abb. 1: Beispiel für ein Entity- Relationship- Modell

Das Entity- Relationship- Modell wirkt als logisch zwingendes Instrument bei der Formulierung von Objekten und Zusammenhängen, die beim Erfassen von Ausschnitten der objektiven Realität erkannt werden. Das Modell dient im Verein mit Begriffskatalogen der entwurfsbegleitenden Dokumentation, die z. T. schon rechnergestützt mit Graphik- Unterstützung erarbeitet wird / 8 /. Das als Entwurfsergebnis vorliegende ERM erlaubt jedoch noch nicht ohne weiteres die Ableitung der äquivalenten Menge von Abhängigkeiten oder gar von Relationen, die in Coddscher Normalform sind. Selbst wenn die Beschreibungen der Gegenstände als Relationen in zweiter oder sogar vierter Normalform vorgegeben und eingehalten werden, können sich immer noch über die dargestellten Beziehungen Verletzungen der Normalformen bei der Ableitung der Beziehungsrelationen ergeben.
So ist z. B. die Beziehung Person - Abteilung in Abb. 1 transitiv.
In der Literatur / 3 / wird dem ERM vorgeworfen, daß es nicht Basis für die Entwicklung automatisierter Entwurfsverfahren sein könne. Vom Verfasser durchgeführte Untersuchungen / 6 / ergaben, daß das ERM sehr wohl Ausgangspunkt für den Ansatz von Algorithmen sein kann, welche die Beziehungsmenge in eine Menge von Abhängigkeiten zerlegen. Diese Abhängigkeiten werden unter Einwirkung der von Armstrong / 9 / aufgestellten und von Beeri, Fagin und Howard / 10 / weiterentwickelten Axiomatik so manipuliert, daß sie Grundlage für eine Datenstruktur sein können, die hohe Datenunabhängigkeit, Redundanzarmut und Beherrschung des Update garantiert. Eine solche Datenstruktur kann als Bestandteil der konzeptuellen Ebene im Sinne der bekannten ANSI/SPARC- Architektur aufgefaßt werden.

3. Transformation des Entity- Relationship- Modells in einen Graphen

Ein graphentheoretischer Ansatz für die Darstellung und Behandlung funktionaler Abhängigkeiten wurde u. a. von Rogge und Tönjes / 4 / verwendet. Neuere theoretische Untersuchungen zur Manipulation funktionaler Abhängigkeiten durch Graphenalgorithmen

liegen von Ausiello, D'Atri und Sacca / 11 / vor.
In diesen Ansätzen werden vorhandene Mengen funktionaler Abhängigkeiten bestimmten Ordnungsprinzipien unterworfen (Normalisierung). Bei einer Übertragung des ERM in ein Graphenmodell steht aber zunächst einmal die Ableitung von Abhängigkeiten aus der Semantik des dargestellten Modells im Vordergrund. Da die funktionale Abhängigkeit ein Spezialfall der mehrdeutigen Abhängigkeit ist, muß im Interesse der Verallgemeinerung und besserer Modellierbarkeit ein Transformationsansatz gefunden werden, der die Ableitung mehrdeutiger Abhängigkeiten gestattet. Das Auftreten mehrdeutiger Abhängigkeiten ist mit der vierten Normalform gekoppelt und kann für den Faginschen Dekompositionsansatz / 1 / ausgenutzt werden.
Abb. 2 zeigt den durch Transformation des ERM nach Abb. 1 erzeugten Graphen. Alle Abhängigkeiten sind in ihrer allgemeinen Form als mehrdeutige Abhängigkeiten durch Doppelpfeile dargestellt.

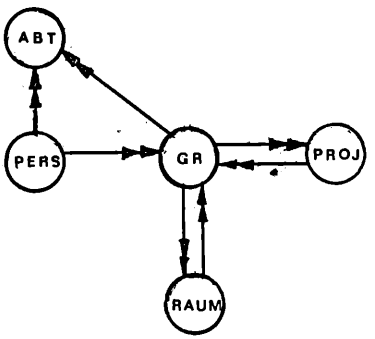


Abb. 2: Durch Transformation erzeugter Graph des Beispiels aus Abb. 1

Bei der Transformation müssen die unterschiedlichen Beziehungstypen und evtl. vorhandene Beziehungsattribute berücksichtigt werden. In Abb. 2 wird der Ersatz der M : N- Beziehungen des ERM durch je einen hin- und rückwärtsgerichteten Doppelpfeil deutlich.

4. Auswertung des transformierten Graphen

Der aus dem ERM transformierte Graph wird als Darstellungsmodell der entworfenen Datenstruktur (Entwurfsschema) einer Behandlung unterzogen, deren Ziel die Herstellung eines verbesserten Schemas ist. Dazu wird der Graph den im folgenden beschriebenen Schritten unterworfen.

Schritt 1:

Ermittlung der stark zusammenhängenden Knoten des transformierten Graphen. Die so gewonnenen stark zusammenhängenden Teilgraphen (zyklische Teilgraphen) geben die Mengen aller durch M : N- Beziehungen miteinander verbundenen Entities an.
Sie sind durch beziehungsdarstellende "Allschlüssel"- Relationen beschreibbar.

Schritt 2:

Bildung eines Graphen ohne zyklische Knotenverbindungen (azyklischer Graph) durch Umwandlung der starken Teilgraphen in Knotenpunkte des azyklischen Graphen.

Schritt 3:

Erzeugung der nichtredundanten minimalen Überdeckung durch Entfernen redundanter transitiver Kanten im azyklischen Graph.

Schritt 4:

Zerlegen der aus den zyklischen Teilgraphen entstandenen Knoten des azyklischen Graphen durch Ermittlung der Knoten des zyklischen Teilgraphen, die zu (nichttrivialen) mehrdeutigen Abhängigkeiten in der durch den Teilgraphen repräsentierten Relation zusammentreten. Für die Erleichterung dieser Prüfung sind

die redundanten Kanten des starken Teilgraphen zu entfernen. Bezogen auf den in Abb. 2 dargestellten transformierten Graph ergibt die Prüfung auf das Enthaltensein zyklischer Teilgraphen gemäß Schritt 1 den durch die Knotenmenge mit den Elementen "GR", "PROJ" und "RAUM" gebildeten starken Teilgraphen TG

$$TG = \{GR, PROJ, RAUM\}.$$

Schritt 2 dient der Bildung eines azyklischen Graphen, der den Teilgraph TG als einen Knoten aufnimmt (s. Abb. 3).

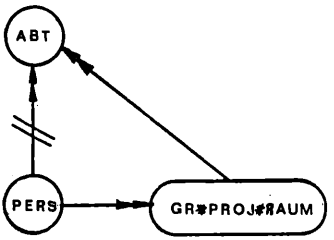


Abb. 3: Azyklischer Graph

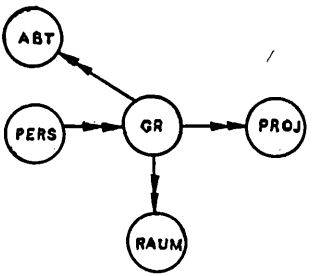


Abb. 4: Resultierender Graph

Gemäß Schritt 3 werden redundante transitive Abhängigkeiten entfernt (gestrichene Verbindung PERS - ABT in Abb. 3). Dieser Schritt ist identisch mit der Herstellung der nichtredundanten minimalen Überdeckung der Abhängigkeiten, die eine starke Verwandschaft zur dritten Normalform aufweist (s. a. / 12 /).

Der in Abb. 3 dargestellte Knoten GR * PROJ * RAUM symbolisiert eine Beziehungsrelation

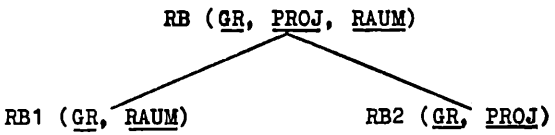
$$RB\ (\underline{GR},\ \underline{PROJ},\ \underline{RAUM})$$

Da sich die Spaltenwerte "RAUM" in dieser Relation für alle Spaltenwerte "PROJ" wiederholen (die "GR" zugeordneten Werte "RAUM" sind unabhängig von "PROJ") wird in RB eine mehrdeutige Abhängigkeit

$$GR \twoheadrightarrow RAUM$$

gehalten. Damit ist RB nicht in vierter Normalform. Auf Basis des Faginschen Dekompositionsansatzes ist eine Zerlegung in

zwei Relationen RB1 und RB2 möglich:



In den resultierenden Graph (s. Abb. 4) sind die in vierter Normalform bestehenden Relationen RB1 und RB2 als mehrdeutige Abhängigkeiten eingebracht. Die Knoten dieses Graphen sind Entityrelationen (Gegenstandsrelationen), seine Kanten stellen Beziehungsrelationen dar. Die Gegenstandsrelationen repräsentieren dabei funktionale Abhängigkeiten eines jeden Attributes zum Schlüsselattribut und sind damit wie die Beziehungsrelationen in vierter Normalform.

Das in den Abb. 1 bis 4 dargestellte Demonstrationsbeispiel ist sehr einfach gewählt. Die Schritte 1 bis 4 können hier ohne weiteres manuell abgearbeitet werden. Bei komplexeren ERM ist dies jedoch nicht so einfach möglich. Für solche Fälle müssen Algorithmen bereitgestellt werden.

5. Algorithmen für die Auswertung transformierter Graphen

Der aus dem ERM transformierte Graph wird zweckmäßigerweise durch graphentheoretische Algorithmen ausgewertet. Für die Realisierung dieser Algorithmen lassen sich vielfach allgemein bekannte Softwarekomponenten verwenden.
Es wurden im einzelnen entwickelt (s. a. / 6 /):

- Algorithmus zum Existenznachweis starker Teilgraphen in einem Graphen

- Algorithmus zur Bestimmung der Knotenmengen starker Teilgraphen auf der Basis von an der Hauptdiagonalen verdichteten Binärmatrizen

- Algorithmus zur Entfernung überflüssiger Kanten in zyklischen Graphen auf Grundlage der Rückführung der Aufgabe auf das Traveling- Salesman- Problem unter der Berücksichtigung

Hamiltonscher Kreise (Anwendung des Programmpaketes DISKO / 13 /).

- Entfernung überflüssiger Kanten in azyklischen Graphen auf der Basis mehrfacher Matrizenmultiplikation

- Prüfung auf das Halten angesetzter mehrdeutiger Abhängigkeiten in Relationen.

Mit Hilfe der angeführten Algorithmen ist eine Rechnerunterstützung für das Umsetzen umfangreicher und in ERM- Form dargestellter Modelle möglich. Sie stellen eine Ergänzung der rechnergestützten Hilfsmittel für den Datenbankentwurf dar, wie sie etwa mit Begriffskatalogen oder Data Dictionary- Systemen zur Verfügung stehen.

6. Zusammenfassung

Das Entity- Relationship- Modell ist ein semantisch reichhaltiges Modell, das eine klare Darstellung der Objekte eines Sachgebietes und der zwischen ihnen bestehenden Beziehungen zu geben vermag. Das dargestellte Modell ist erweiterungsfähig. Auf Grund der genannten Eigenschaften ist das ERM für den Datenbankentwurf aus dem Sachgebiet heraus gut geeignet.
Die Transformation des ERM in einen Graphen und dessen analytische Behandlung erleichtern eine rechnergestützte Übernahme der Vorstellungen des Entwerfers (Sachgebietsspezialisten) durch den EDV- Fachmann in der Form abgeleiteter Abhängigkeiten. Für die Analyse des transformierten Graphen wurden Algorithmen entwickelt, die weitgehend auf allgemein eingeführter Software beruhen. Die Algorithmen liefern eine Struktur von Relationen in vierter Normalform für das betrachtete System.
Die entwickelten Algorithmen stellen somit eine Ergänzung der Anwendung des Entity- Relationship- Modells im Sinne der Überführung von Modellvorstellungen in logische Datenbankstrukturen dar.

Literaturverzeichnis

1 Fagin, R.: The Decomposition versus the Synthetic Approach to Relational Database Design. In: Proc. 3rd VLDB, Tokyo 1977, S. 441-446.

2 Tsichritzis, D.C.; Lochovsky, F.H.: Designing the Data Base. Datamation, New York 24(1978)8, S. 147-151.

3 Krieger, R.; Lausen, G.; Stucky, W.: Methodischer Datenbankentwurf beim Aufbau betrieblicher Informationssysteme. Angew. Informatik, Braunschweig 24(1982)3, S. 191-198.

4 Rogge, R.; Tönjes, J.: Strukturierung von Datenbanken im Relationenmodell. Inst. für Energetik, Leipzig 1979.

5 Chen, P.P.S.: The Entity- Relationship- Model - Toward a Unified View of Data. ACM Transactions on Database Systems, New York 1(1976)3, S. 9-36.

6 Prescher, G.: Zur Projektierung von Anwenderdatenbanken für die Lösung informationeller Aufgaben in der Elektroenergiewirtschaft. Entwurf Diss. A (unveröffentlicht).

7 Chen, P.P.S.: Entity- Relationship Approach to Systems Analysis and Design. Amsterd./ New York/ Oxford 1980.

8 Chan, E.P.F.; Lochovsky, F.H.: A Graphical Data Base Design Aid using the Entity- Relationship- Model. In: Chen, P.P.S.: Entity- Relationship-Approach to Systems Analysis and Design S. 295-310. Amsterd./ New York/ Oxford 1980.

9 Armstrong, W.W.: Dependency structures of data base relationships. In: Proc. of the IFIP 1974, S. 580-583.

10 Beeri, C.; Fagin, R.; Howard, J.H.: A complete Axiomatization for Functional and Multivalued Dependencies in Database Relations. In: Proc. ACM- SIGMOD Conference, S. 47-61, Toronto 1977.

11 Ansiello, G.; D'Atri, A.; Sacca, D.: Graph Algoritms for Functional Dependency Manipulation. Journal of the ACM, New York 30(1983)4, S. 752-766.

12 Wedekind, H.: Datenbanksysteme I. Mannheim/ Zürich 1974.

13 Programmpaket DISKO, PP für die Lös. von Probl. der diskreten Optimierung, VEB Komb. Robotron. Dresden 1977.

Verfasser:

Dipl.- Ing. G. Prescher

Ingenieurhochschule Zittau

Abt. EDV und Rechentechnik

8800 Zittau

Theodor- Körner- Allee 16

Rácz, Éva; Horváth, János; Kozma, László

ON TESTING OF A DISTRIBUTED DATABASE MANAGEMENT SYSTEM

Abstract

A method has been developed for testing of a process control database management system. The database is distributed on various mini- and microcomputers interconnected by a high-speed communication network. The aim of the testing was

- to check functional correctness of the system,
- to set up an installation test system,
- to measure the most important characteristics of the system.

This paper describes the applied methods and software tools.

1. Introduction

1.1. The PCDB system

The PCDB (Process Control Database Management System) [3] allows users to perform various operations on a distributed database, where the data are distributed amongst local bases stored on mini- and microcomputers interconnected by a high-speed communication network. The minis are TPA 11 computers supervised by the operating system RSX 4.0, the micros are INTEL 8080 based intelligent (CAMAC) crate controllers (ICC-s) supervised by the minis.

The components of the PCDB are executed by the TPA 11 minicomputers except the program handling the data residing on the ICC-s.

The PCDB makes use of the relational data model. Data can be stored in three places: in the memory or on the disks of a minicomputer or in the memory of a microcom-

puter.

There are three types of relations: relative, special and indexed relations. Relations residing in the core can have one of the following two types: relative or special. Relations stored on disks can be indexed in addition. One relation can be resident only on one medium at a time.

Thre are four types of relation manipulation routines [5]. The first group consists or routines reserving and releasing relations. The record manipulation routines operating on the rows of a relation form the second group. The third group of the routines operates on the fields (attributes and bittributes) of a relation and the fourth one operates on the columns of a relation. The routines of the second and the fourth group can be executed in synchronous or asynchronous way.

The schema describing the relations can be stored in the memory or on the disk of the local minicomputer controlling the corresponding data. In the schema one relation is represented by one record. The schema description retrieval routines give informations about the relations or about the fields of a relation.

1.2. The test system

The test system of the PCDB has been designed in cooperation with the constructors of the PCDB on the basis of the PCDB user's manuals.

Our task was to create a test system complying with the following principal requirements:

- checks the functional correctness of the services supported by the PCDB for users,
- maintains the measuring of the most important characteristics of the system,
- can be used as the test system of a new installation.

Applying Hetzel's terminology [1] our aim was to set up a test system supporting the so-called certification and performance procedures.

The further part of the paper discusses briefly the general principles of the test method (section 2). In the third section there are presented the software tools used during the working up of the test system. The important characteristics are discussed as well. The fourth section gives a summary of our experiences.

2. General pricinples

The test system consists of two main parts, one of them are the programs and the other is the database.

Our requirements concerning the database were as follows:

- the database should contain at least one relation of all kinds on storages of the possible types,
- the internal structure of the relations should make it possible to test the normal working of the PCDB and to check the error messages,
- the database should support the examination of the synchronization and effectiveness of the distributed system,
- the same database should be loadable on each mini-computer of the network,
- the loading of the data sets of the database should be recovered easily after a system crash.

The programs form more subsystems according to the relation manipulation routines:

- open-close subsystem to test the reservation and release of the relations,
- record manipulation subsystem to test the record management routines,
- field manipulation subsystem to test field operations,
- column manipulation subsystem to test column operations,
- schema description subsystem to test schema description retrieval operations.

There is a further subsystem, the so-called performance analysing subsystem to measure the most important features of the PCDB. Each subsystem consists of one or more programs written on the programming language Fortran IV. The programs are connected with each other by command files to form a subsystem. The subsystems can be executed independently, but can be arranged as a single test system by means of command files. Several copies of the test system can be executed with different parameters on the different minicomputers of the network at the same time.

3. Applied tools

3.1. Software tools

Several utility programs are attached to the PCDB system to support the construction, loading and modification of relations [4].

For example,

a) the schema processor generates the following output files

- the internal format of the schema,
- source listing with error messages,
- RMS indexed file definitions,
- FCS random access file definitions for relative relations;

b) relation editor (RDT) is an interactive utility program for loading, modifying and testing the fields of the records;

c) relation copy program (RCP) is a utility for copying a sequential file in character form into a relation and inversely. The sequential file can be generated and modified by any text editor.

The operating system and the PCDB system itself have a number of services to query the states of the running programs and the relations used by them.

One of the most important services of the PCDB is the Monitor. The state of the database can be examined by the Monitor at optional intervals. For example, the mode of reservations, the user of the relation can be inspected. Informations can be asked about the free space of the spooler, about the contents of the tables of the PCDB.

3.2. The test database

The test database has been constructed according to the principles mentioned above. The database consists of about 50 relations, 42 relations of them have been divided into 6 relation-groups. This is done to simplify the loading of the database.

The groups have been composed from the relations having the same record structure apart from the "keyed" attribute.

As a relation can have relative, special or indexed organization and can be stored on the disks or in the core of the computers (except indexed relations wich can be stored on the disks only), seven relations belong to a group.

The relations can be specified for the schema processor as follows:

```
RELATION RDSPDP
  (RSQN : INT2, KEYED: 1, 2;
   KEY1 : STR, KEYED : 3, 6;
   KEY2 : STR, KEYED : 9, 2;
   VFL4 : FLT4 VECTOR: 11, 40)
```

```
ALLOCATION: 15 RECORDS;
SPECIAL;
SCHEMA: DISK; DATA: DISK;
DATA FILE: 'AB = [215, 216] RDSPDP.SPC'
```

The relation above declares a relation of special type. Each relation has a key named RSQN (record sequence number). In addition it has a key-field of string type and a key-field of integer type. Data-fields are floating-point numbers forming a vector of 10 elements. The relation can have 15 records at most. The schema describing the relation and the relation itself are stored on the disk. The relation is saved in the standard DOS-RV file named RDSPDP.SPC.

The relation residing in the core and belonging to the same group as the relation above can be declared as follows:

```
RELATION RDRECP
  (RSQN: INT2, KEYED: 1, 2;
   KEY1: STR        : 3, 6;
   KEY2: INT2       : 9, 2;
   VFL4: FLT4 VECTOR: 11, 40)
ALLOCATION: 15 RECORDS;
RELATIVE;
SCHEMA: DISK; DATA: CORE;
DATA FILE: 'AB = [215, 216] RDRECP.COR'
```

Specifying the relation-groups in the way outlined above, it is sufficient to create a single copy of the source files for each group, and the relations can be loaded from them by means of the utility RCP.

The structure and the content of the relations have been fixed according to the services tested by the group of the relations.

The test database can be loaded on several minis of the network at the same time, moreover while testing the distributed system the database must be loaded on all of the minis.

3.3. The test programs

The programs have been arranged into subsystems according to the operations on the relations. An optional system of programs can be formed from the subsystems by command files. Several copies of the systems can be executed on the minis at the same time. Parameters can be passed to the systems by different DOS-RV files. The files of the parameters can be created by command files.

The parameters can be given in the following way:

- residence type: ALL, DISK, CORE or ICC.
  It defines the storage containing the relations to be tested;
- mode of execution: ALL, SYNCHRONOUS or ASYNCHRONOUS.
  It defines the version of the record management routines to be tested;
- number of processors: a number of integer type.
  It defines the number of minicomputers to be tested.
- processor identifier 1: identifier of a minicomputer;

  ⋮
- processor identifier n: as above.

The test programs begin their works according to the parameters with the database on the local computer. The same activity will be executed on each given minicomputer of the network after each other.

To save the consistency of the database, the test programs modifying the relations must recover the database.

This feature of the programs makes it possible to execute several copies of the test system at the same time and to use the database without loading again supposing there is no other reason (for example system crash).

4. Experiences

4.1. Errors detected during organization

The test system has been constructed on the basis of the user's documentations. The error messages of the PCDB proved to be incomplete. Seven new messages have been created. We directed attention of the constructors to the incompletness and the ambiguity of the documentations (about 10 instances).

4.2. Errors detected during creation of the database

Using the schema processor five errors have been discovered, each of them can be led to the carelessness of the constructors.

For example, there were given no error messages when bittributes were defined to the vector attributes, or host attributes were declared as keyed variables (both are prohibited).

4.3. Errors detected during programming

About 25 errors were detected during the testing of the record management routines. Three of them can be led to planning problems, the others proved to be errors of the coding. Several errors were due to the poor logical checking of the field and column operations. The source of three errors turned out to be the faulty sinchronization of the distributed system.

4.4. Performance analysis

The performance analysing subsystem has been created to measure the most important features of the distributed system. The subsystem makes it possible to measure the average execution time of certain operations on different relations both in the local and in the distributed system under given software and hardware circumstances.

The complete performance analysis of the system will be made in the future, but there are already a number of remarkable conclusions.

For example, the searching time should be incependent of the place of the records in the case of relations of special type residing in the core. Whereas it was found, that the searching time increased highly in the case of local relations by the growth of the RSQN. On the other hand, in the case of remote relations the time increased in a substantially smaller degree. Obviously the searching algorithm must be improved in the local case.

Literature

[1] Ed. by Hetzel, W. C.: Program test methods. London, Prentice-Hall, 1973.

[2] Várkonyi, Zs.: Bevezetés a modern programtesztelésbe. Müszaki Könvvkiadó, 1979. (in Hungarian).

[3] PCDB Overview. Manuscript.

[4] PCDB Utilities. Manuscript.

[5] PCDB RMR routines. Manuscript.

Authors

Rácz, Éva
Computer Centre of L. Eötvös University
H-1117. Budapest, Bogdánfy u. 10.

Horváth, János
Kozma, László
Department of Computer Science
L. Eötvös University
H-1088. Budapest, Muzeum krt. 6-8.

САМАРИН М.К.

Автоматизированная система управления Московским университетом

В сообщении рассматриваются общие принципы организации и функционирования автоматизированной системы управления Московского университета, используемой для обеспечения учебно-научной, административной и финансово-хозяйственной деятельности МГУ. Дается описание архитектуры системы, языков общения пользователей с АСУ, программных средств организации и ведения баз данных. Особое внимание уделяется проблеме унификации проектных и технологических решений при создании функциональных подсистем АСУ МГУ.

Trautloft, R.

Spezifikation von Software zur Verwaltung von Datenbasen

1. Einführung

Ein wichtiger Gegenstand der Softwaretechnologie ist die Beschäftigung mit der Entwicklung von Softwaresystemen, die zur Automatisierung und Rationalisierung komplexer informationeller Prozesse geschaffen werden. Kennzeichen solcher Softwaresysteme ist eine umfangreiche, meist komplex strukturierte, permanent zu speichernde Datenbasis. Der "Stand der Kunst" ist dabei u.a. gekennzeichnet durch geeignete Modularisierungsprinzipien, die sich auf das PARNAS'sche "information hiding" zurückführen lassen. Eine direkte Anwendung solcher Prinzipien ist die Betrachtung der Datenbasis und der auf ihr erlaubten Operationen als abstrakter Datentyp bzw. Datenkapsel /TRAU 84/.
Im weiteren soll davon ausgegangen werden, daß das zu entwickelnde Softwaresystem so entworfen ist, daß Datenbasismanipulationen auf einem von physischen Details unabhängigen Niveau beschrieben sind. Ein entsprechendes sprachliches Mittel zur Erreichung dieses Ziels ist die Entwurfssprache PADEL /LITK 84/. Dann können die in der Implementierungsphase anstehenden Aufgaben wie folgt kurz umrissen werden:

- Überführen des Entwurfsergebnisses in die Notation der Implementierungssprache,
- Implementieren der Datenbasis und der Datenbasissoftware.

Die zweite Aufgabe steht im Mittelpunkt der weiteren Ausführungen. Die Implementation der Datenbasis soll die folgenden Aktivitäten umfassen:

- Treffen von Implementierungsentscheidungen über die physische Repräsentation der Objekte der Datenbasis /TRAU 82a, TRAU 82b/,
- Anlegen der Datenbasis,
- Laden der Datenbasis.

Mit "Datenbasis-Software" wird zunächst die programmtechnische Realisierung der Operationen auf den Objekten der Datenbasis bezeichnet. Es wird noch sichtbar, daß dies nur ein minimaler Kern der Datenbasissoftware ist. Sie zu implementieren ist nicht trivial. Bei modellhafter Betrachtungsweise läuft daher im Softwareentwicklungsprozeß (Prozeßprodukt: Softwaresystem zur Automatisierung informationeller Prozesse) ein weiterer Softwareentwicklungsprozeß (Prozeßprodukt: Datenbasissoftware) ab.

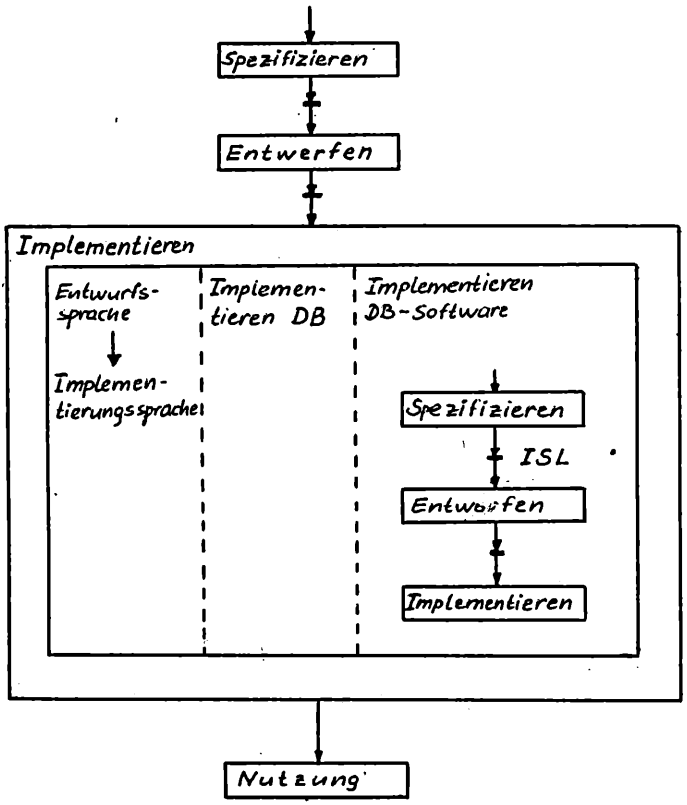


Abb. 1

Wichtig ist die folgende Forderung: Im Ergebnis des Spezifierens entsteht eine Spezifikation. Ihr Inhalt muß hinsichtlich Vollständigkeit und Genauigkeit so beschaffen sein, daß die zu entwickelnde Datenbasis-Software alle notwendigen Funktionen bei Einhaltung erforderlicher Leistungsparameter erfüllt.

2. Inhaltliche Anforderungen an die Spezifikation von Datenbasis-Software

Bezüglich des Leistungsvermögens von Datenbank-Software muß eine Orientierung an modernen Datenbankbetriebssystemen (DBMS) erfolgen, denn diese sind nichts anderes als universell nutzbare Datenbasis-Software. Daraus folgt, daß die inhaltlichen Anforderungen an die Spezifikation von Datenbasis-Software aus den folgenden Fähigkeiten abzuleiten sind:

(1) Hauptaufgabe jeder Datenbasis-Software muß es selbstverständlich sein, den Zugriff zu den Daten der Datenbasis zu gewährleisten. Dazu ist die Abbildung der Objekte auf logischer Ebene in physische Objekte zu handhaben.

(2) Da möglicherweise nicht jeder Nutzer zu allen Daten zugreifen darf bzw. unterschiedliche Manipulationen für unterschiedliche Nutzer erlaubt sind, muß die Zugriffskontrolle an Hand bestimmter Autorisierungsrechte ausgeübt werden.

(3) Punkt (2) spricht bereits die Existenz mehrerer Nutzer an. Im allgemeinen Fall arbeiten einige davon im konkurrierenden Modus, so daß auf der Basis geeigneter Sperreinheiten ein Sperrmechanismus zu realisieren ist.

(4) Ohne hier die verschiedenartigsten Definitionen der Integrität diskutieren zu wollen, steht doch fest, daß im weitesten Sinne Maßnahmen für die Wahrung der Integrität der Datenbasis zu realisieren sind.

(5) Die Datenbasis-Software hat Recovery-Maßnahmen zu gewährleisten.

(6) Wie ein roter Faden zieht sich durch alle bisher genannten Fähigkeiten bzw. Aufgaben: die Forderung nach Effizienz. Um Effizienz an der einen oder anderen Stelle zu erreichen, führen deshalb moderne, leistungsfähige DBMS bestimmte Optimierungen durch.

(7) Eng im Zusammenhang zu den Optimierungen, aber auch als Basis für Tuningmaßnahmen, stehen Messungen bzw. Sammlung statistisch auswertbarer Informationen.

(8) Zunehmend wird Bedeutung erlangen, daß einzelne Funktionen der Datenbasis-Software durch Hardwarekomponenten realisiert werden können.

3. Zur Exaktheit der Spezifikation

Es ist viel diskutiert worden zu Pro und Contra von Formalisierungen auf der Ebene der Spezifikation, z.B. /KREO 81/. Folgende Überlegungen sind bei einer Einordnung zwischen die beiden Extremwerte "natürlich-sprachlich" und "formal" im hier vorliegenden Fall anzustellen:

- Eine Rechnerunterstützung beim Treffen vom Implementierungsentscheidungen /TRAU 82a, TRAU 82b/ kann dazu benutzt werden, Teile der Spezifikation (Punkt (1) im Abschn. 2 betreffend) maschinell zu erzeugen.
- Eine exakte Spezifikationssprache bietet darüber hinaus potentiell die Möglichkeit der maschinellen Weiterverarbeitung, was einer Rechnerunterstützung der nachfolgenden Phasen des Softwareentwicklungsprozesses entspricht.

Um diesen Anforderungen gerecht zu werden, wurde eine Spezifikationssprache (ISL) konzipiert, die sowohl maschinell verarbeitbar als auch durch hohe Problemtransparenz und deskriptives Niveau gut vom Softwareentwickler handhabbar sein soll.
Entsprechend der in Abschn. 2 aufgelisteten inhaltlichen Anforderungen sind zugehörige ISL-Sprachteile konzipiert, wobei diese Aussage nicht bedeutet, daß es sich dabei um selbständige, separierbare Teile handelt.

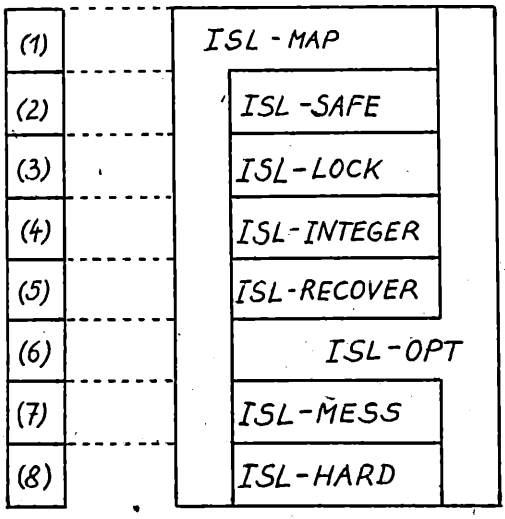


Abb. 2

Neben der Zuordnung der inhaltlichen Anforderungen zu Sprachteilen soll Abbildung 2 die grundsätzliche - weil alles tangierende - Bedeutung von ISL-MAP und ISL-OPT zeigen.
Die konkretesten Vorstellungen existieren gegenwärtig bezüglich ISL-MAP (s. unten), die anderen Teile sind ganz grob "angedacht". ISL-HARD stellt einen "weißen Fleck" dar.

4. Einige ausgewählte Bestandteile von ISL-MAP

Entsprechend der Aufgabe von ISL-MAP, nämlich die Zuordnung von logischen Datenbasis-Objekten und deren Manipulation zu physisch existierenden Objekten und deren Manipulation zu gewährleisten, muß ISL-MAP die folgenden Bestandteile enthalten:

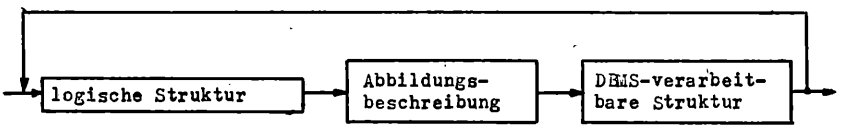
(1) Abbildungsbeschreibung

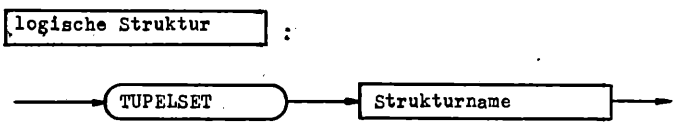
(2) Beschreibung der logischen Datenbasis-Objekte (TUPELSET)

(3) Beschreibung der physischen Datenbasis-Objekte (TABLE, LINKTABLE)

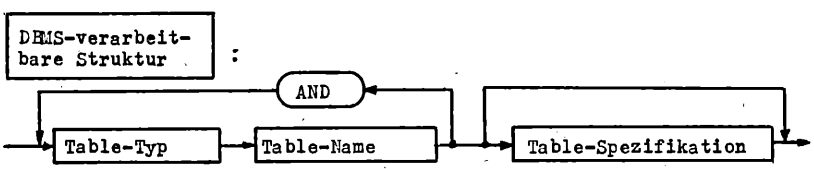
(4) Beschreibung der Attribut-Typen der Datenbasisobjekte.

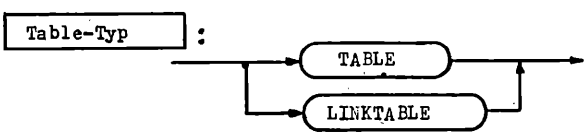
Einige Bemerkungen zu (2):





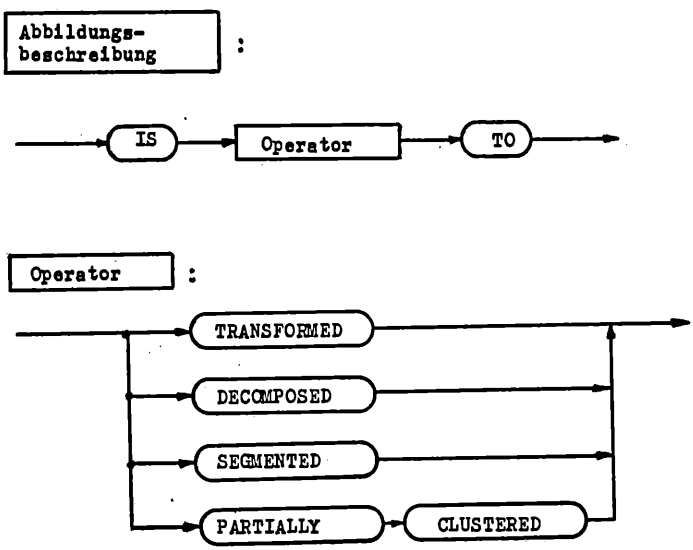
Auf der physischen Ebene gibt es zwei Objekttypen, das TABLE und das LINKTABLE, das sich nur hinsichtlich seiner Aufgabe vom TABLE unterscheidet. Mit ihm soll angezeigt werden, daß eine Verbindung zwischen TABLEs existiert. Diese Information kann in Abhängigkeit von den verfügbaren Speicherstrukturen für die Implementation ausgenutzt werden. Die weiteren Angaben im nachfolgenden Syntaxdiagramm sind abhängig von der Abbildungsbeschreibung, die unten noch vorgestellt wird.





Für die syntaktische Notation der Abbildungsbeschreibung sind zunächst mögliche respektive sinnvolle Abbildungen zu betrachten. Die einfachste Form ist eine 1:1-Abbildung in dem Sinne, daß ein logisches Objekt (TUPELSET) auch ein physisches Objekt (TABLE oder LINK-TABLE) wird. Dafür wurde der Operator TRANSFORMED geschaffen. Eine weitere Möglichkeit besteht darin, das logische Objekt aufzuteilen durch eine horizontale Dekomposition (Operator DECOMPOSED) oder durch eine vertikale Dekomposition (Operator: SEGMENTED). In beiden Fällen entstehen mehrere physische Objekte (durch AND aneinandergereiht), und in der 'Table-Spezifikation' wird beschrieben, was diese Objekte voneinander unterscheidet.
Für das vollständige oder teilweise Zusammenfügen von logischen Objekten zu einem physischen wurde der Operator PARTIALLY CLUSTERED eingeführt.

Es ergibt sich:



Ein kurzes Beispiel illustriert die bisher eingeführten Konstruktionen.

Beispiel:

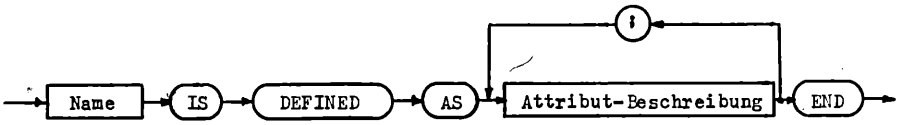
TUPELSET material IS TRANSFORMED TO TABLE material

TUPELSET disp IS TRANSFORMED TO TABLE disp

TUPELSET matdisp IS CLUSTERED TO TABLE material

TUPELSET matlag IS TRANSFORMED TO LINKTABLE matlag

Für die detailliertere Beschreibung der Datenbasis-Objekte ((2) und (3)) ist weitgehend übereinstimmend konzipiert:



Hinzu kommt beim LINKTABLE die Beschreibung des Links (der Verbindung). Innerhalb der Attributbeschreibung ist im wesentlichen zu spezifizieren, ob es sich um ein Primärschlüsselattribut, ein Linkattribut, ein Sekundärschlüsselattribut handelt.

5. Äquivalenzen zum Datenbankentwurfsprozeß und Schlußfolgerungen

Im Abschnitt 1 erfolgte die Einordnung der Datenbasis-Software-Entwicklung in einen übergeordneten Softwareentwicklungsprozeß. Dies war nützlich für die Herausarbeitung der Spezifikationsschnittstelle. Betrachten wir nun den Datenbanklebenszyklus und fragen, ob es dort Verwendung für die (oder eine ähnlich) konzipierte sprachliche Schnittstelle gibt.

Nach der Phase des Entwerfens im Softwareentwicklungsprozeß ist der Beschreibungszustand der Datenbasis vergleichbar mit dem konzeptuellen Schema einer Datenbank (entspr. ANSI/SPARC-3-Ebenen-Konzept). In der Implementierungsphase des SEP ist dann eine Abbildung auf die physischen Gegebenheiten einer Basismaschine zu realisieren, was vergleichbar mit der Aufgabe des Datenbankadministrators (DBA) ist, das konzeptuelle Schema in ein internes Schema zu transformieren.

Zwei Dinge sind an dieser Stelle bemerkenswert:

(1) Voraussetzung dafür, daß der DBA hierbei vor einer anspruchsvollen Aufgabe steht, ist eine gewisse Leistungsfähigkeit des zu nutzenden DBMS, d.h., daß der DBA überhaupt Entscheidungsmöglichkeiten zur Auswahl physischer Realisierungen besitzt.

(2) Während auf konzeptueller und externer Ebene sprachliche Mittel zur Verfügung stehen, gibt es auch international keine Möglichkeit der exakten Beschreibung des internen Schemas und seiner Beziehung zum konzeptuellen Schema. (Eine Ausnahme bildet der CODASYL-orientierte Ansatz, beschrieben in /TOZE 78/).

Dabei wäre eine solche exakte sprachliche Schnittstelle mehrfach bedeutsam:

(1) Sie wäre das Instrument des DBA zur Definition des internen Schemas.

(2) Sie könnte für den DBA Instrument und Bezugspunkt für Tuningmaßnahmen sein.

(3) Sie könnte Dokumentationszwecken im Datenbankentwurfsprozeß dienen.

(4) Beschreibungen auf dieser sprachlichen Ebene wären abspeicherbar und maschinell interpretierbar, so daß eine Basis für die durch das DBMS zu realisierende Abbildung ("schema mapping") vom konzeptuellen in das interne Schema gegeben wäre.

6. Konzeptionelle Überlegungen für die Rechnerunterstützung des Datenbankentwurfsprozesses

Die nachfolgende Abbildung 3 soll zunächst graphisch die Konsequenz der bisherigen Überlegungen veranschaulichen. Die geschwärzten Verbindungen dokumentieren, daß das DBMS für seine Arbeit auf jeden Fall Informationen über das konzeptuelle und das interne Schema sowie deren Zusammenhang braucht. Die Frage ist nur,

- wie umfangreich diese Informationen sind,
- wie sie verfügbar gemacht werden können,
- wofür sie noch genutzt werden können.

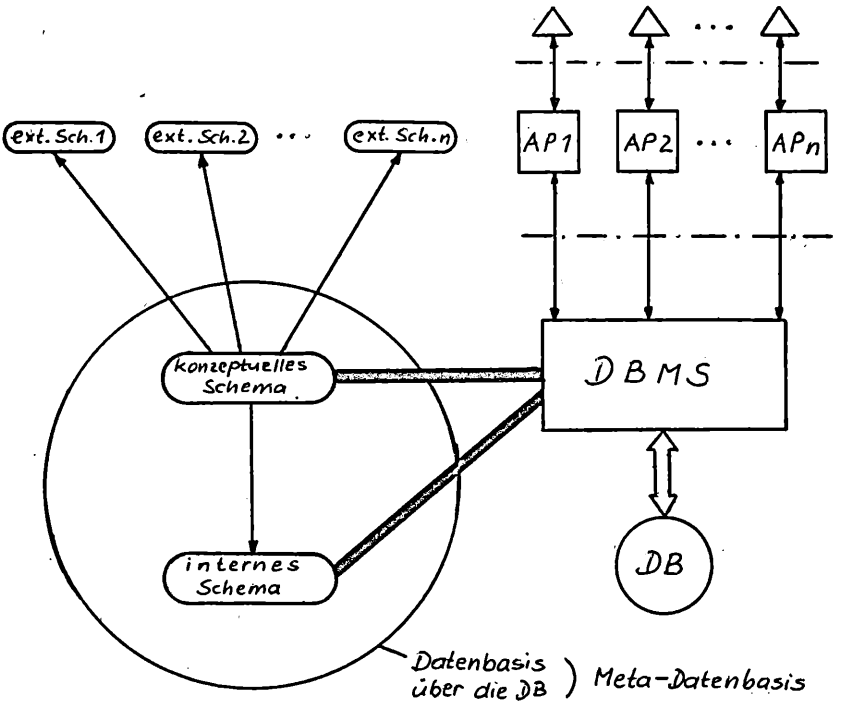


Abb. 3

ext. Sch. = externes Schema; AP = Anwendungsprogramm;
△ = Nutzer bzw. Nutzergruppe

Über die Arbeit des DBMS hinaus könnte damit eine Meta-Datenbasis für den DBA zur Verfügung stehen. Dies drängt den Gedanken an ein Data Dictionary (DD) auf, dessen Aufgabe ganz allgemein darin besteht, Daten über Daten abzuspeichern.
Der internationale Trend zeigt eindeutig /MELL 84/:

(1) die zunehmende Bedeutung (integrierter) Data Dictionaries
und
(2) den Wandel von ihrer ursprünglich ausschließlich deskriptiven Funktion zu einer zunehmend steuernden, den Nutzer (hier: DBA) bei seiner Arbeit führenden Funktion.

Das DD, mit entsprechenden Fähigkeiten ausgestattet, kann zum"Werkzeugkasten des DBA" werden. Mit zunehmendem Integrationsgrad und Erweiterung der Anwendbarkeit auch auf frühe Phasen des Datenbanklebenszyklus (z.B. logischer Datenbankentwurf) führt dann die Entwicklung analog zum Softwareentwicklungsprozeß zu einem Database Engineering Environment (DEE bzw. DE^2 ; vgl. SEE bzw. SE^2 /HESS 81/).

Es gibt bereits Veröffentlichungen /ORR-84/, die - ohne explizite Bezugnahme auf den Datenbanklebenszyklus - noch einen Schritt weitergehen und als Nachfolger die CAD/CAP-Systeme (Computer Aided Design/ Computer Aided Programming) für die nächsten Jahre voraussagen. Deren wesentlichen Merkmale sollen sein:

- Überstreichen eines weiten Wirkungsbereiches (Planungsphase, Spezifizierungsphase, Entwurf und Implementierung, Datenbasisentwurf, Wartung usw.)
- die an diesen Prozessen Beteiligten haben direkten Zugriff zu ihren Datenbasen, um Systeme, Programme, Datenbasen, Dokumente usw. zu manipulieren und zu entwickeln.

Für den "Werkzeugkasten" sind zunächst folgende Inhalte denkbar:

(1) Informationsdienste
Unterschiedliche Informationen (Relationen der DB, Attribute der Relationen, Wertebereiche der Attribute, angelegt (Datum), letztes Hinzufügen, existierende Sekundärzugriffspfade usw.) sind in unterschiedlichen Formen (Reports, Diagramme, grafische Übersichten usw.) ableitbar.

(2) Werkzeug(e) für den physischen DB-Entwurf
Das Instrumentarium, das hier in Betracht kommt, wird zunächst durch die beim physischen DB-Entwurf anstehenden Entscheidungen beeinflußt /TRAU 83/.
Vorrangig wären zu nennen:
- Auswahl von Speicherstrukturen
- Auswahl von Sekundärschlüsseln
- Auswahl von Realisierungsvarianten für die Zugriffspfade an Hand von Sekundärschlüsseln
- Auswahl von Speicherzuordnungsstrukturen.

Dieses Spektrum ist DBMS-abhängig, sowohl hinsichtlich des Umfangs als auch hinsichtlich der einzelnen Entscheidungsräume.
Unabhängig von der konkret anstehenden Entscheidung, läßt sich Rechnerunterstützung typischerweise wie folgt erreichen:
- Generierung von Varianten
- Bewertung von Varianten

und eventuell
- Auswahl einer optimalen Variante bzw. Unterstützung bei der Auswahl einer optimalen bzw. "guten" Variante.

(3) Werkzeug(e) für den logischen Datenbankentwurf (bis zur Formulierung des konzeptuellen Schemas)

(4) Werkzeug(e) zur Sammlung und Analyse von Anforderungen an die DB im Verlaufe der Nutzungsphase.
Die Sammlung von Informationen über die Nutzung der DB und ihre perspektivische Entwicklung ist von großer Bedeutung für den DB-Entwurfsprozeß, unabhängig davon, ob dieser rechnerunterstützt abläuft oder nicht. Die Führung des Entwerfers sowie die Einbeziehung von Nutzern sind durch die angesprochenen Werkzeuge anzustreben.

(5) Tuning-Möglichkeiten für den DBA

7. Literaturverzeichnis

/HESS81/ Hesse, W.: "Methoden und Werkzeuge zur Software-Entwicklung - Ein Marsch durch die Technologie-Landschaft", Informatik-Spektrum 4(1981)4, S. 229-245

/KREO81/ Kreowski, H.-J.: "Wo liegen die Grenzen der praktischen Anwendung formaler Methoden für die Entwurfsspezifikation?", Floyd/Kopetz (Hrsg.): "Software Engineering-Entwurf und Spezifikation", B.G. Teubner, Stuttgart, 1981

/LITK84/ Lindner, U.; Trautloft, R.; Kaiser, U.: "Entwerfen in der pascal-orientierten Entwurfssprache PADEL", INFO'84, Dresden, 6.-10.02.1984

/MELL84/ Mellema, P.M.: "Specification as Data Bases", Int. Werkshop on Models and Languages for Software Specification and Design, 30.03.84, Orlando (Florida)

/ORR-84/ Orr, K.:"A Program for Building Quality Software at High Speed", ebenda

/TOZE78/ Tozer, E.E.: "The Data Storage Description Language (DSDL)", in: Database Technology, State of the Art, Infotech, 1978, Vol. II

/TRAU82a/ Trautloft, R.: "Über einen Zugang zur automatisierten Auswahl physischer Realisierungen von relational definierten Bestandsdaten", bit '82, 28.6.-2.7.1982, Tagungsbericht S. 209 ff.

/TRAU82b/ Trautloft, R.: "Selektion von Implementierungsstrukturen - Ein Automatisierungsansatz für die Implementierung der Datenbasis von Softwaresystemen", edv-aspekte, 1 (1982)3, S. 63 ff.

/TRAU83/ Trautloft, R.: "Über eine Möglichkeit der Rechnerunterstützung bei der Implementierung der Datenbasis von Softwaresystemen für automatisierte Informationssysteme", Diss., TH Karl-Marx-Stadt, 1983

/TRAU84/ Trautloft, R.: "Einige Aspekte der Realisierung physischer Datenunabhängigkeit", Fachtagung "Software für automatisierte Informationssysteme" 17./18.10.1984, Karl-Marx-Stadt

Dr. - Ing. Rainer Trautloft

TH Karl-Marx-Stadt, Sektion Informatik
Straße der Nationen 62

Waldmann, Lutz

KDS2000 - ein datenbankähnliches Datenverwaltungssystem für Mikrorechner K1600

1. Einführung

KDS2000 dient zur Verwaltung kleiner bis mittlerer Datenmengen auf Mikrorechner K1600. Es kann in verteilten Systemen zur temporären Verwaltung von Auszügen großer Datenbestände, die auf zentralen ESER-Rechnern geführt werden, oder zur eigenständigen Verwaltung von Daten eingesetzt werden. Mit KDS2000 ist parallel über mehrere angeschlossene Terminals und einer Stapeleingabedatei ein schneller direkter Zugriff auf die Daten möglich. Die gerätetechnischen Mindestanforderungen (1 Direktzugriffsspeicher, 64 KB Hauptspeicher) erlauben einen Einsatz auf allen in der DDR installierten Mikrorechnersystemen K1600.

Bei der Entwicklung von KDS2000 wurden Erfahrungen aus der Entwicklung des DBBS DAFEMA berücksichtigt, so daß KDS2000 auf einem ausbaufähigen bewährten Grundkonzept basiert. Der logischen Datenstruktur liegt ein relationales Datenkonzept zugrunde. Die interne Ablaufsteuerung erfolgt in Form einer Transaktionssteuerung. Der Anwender kann durch das Programmieren von eigenen Transaktionen KDS2000 fast uneingeschränkt an seine speziellen Probleme anpassen. Bereitgestellte Makros unterstützen den Zugriff auf die Datenbasis und ermöglichen die Inanspruchnahme weiterer Systemdienste. Grundfunktionen wie Zugriffsschutz, Laden und Auslagern eines Datenbestandes und Blättern in bereitgestellten Bildschirmfolgen werden durch vorhandene Transaktionen realisiert. Im Rahmen der Weiterentwicklung von KDS2000 ist die Bereitstellung von allgemein nutzbaren Transaktionen für alle Grundfunktionen der Datenmanipulation auf Satzebene vorgesehen.

Der Einsatz von KDS2000 als Komponente in verteilten Systemen ist in der 1. Ausbaustufe möglich, in dem Daten eines zentralen Datenbestandes ausgewählt und zum Mikrorechner übertragen werden. Dort werden sie unter Steuerung von KDS2000 zum Laden oder Modifizieren einer Datenbasis verwendet.

Die in der weiteren Arbeit vorgenommenen Änderungen oder der geänderte Datenbestand werden zum zentralen Rechner zurückübertragen und dort wieder in den zentralen Datenbestand eingearbeitet.

2. Dateisystem

Für alle von KDS2000 verwalteten Dateien wird FCS als Zugriffsmethode verwendet. KDS2000 verwaltet die Datenbasis in einer Basisdatei und einer Suchdatei. Die eigentlichen Nutzerdaten werden in der Basisdatei gespeichert. Darin können bis zu 16 logische Basisdateien (Datenbestände, Relationen) getrennt verwaltet werden. Gegebenenfalls können in einer logischen Basisdatei auch mehrere Relationen abgespeichert werden. Die kleinste adressierbare Einheit innerhalb der Basisdatei ist der Satz. Die Satzlänge ist variabel und beträgt maximal 500 Byte. Der Aufbau der Basissätze ist vollkommen wahlfrei.

Sollen Datenbestände mit den vorgesehenen allgemein nutzbaren Transaktionen bearbeitet werden, müssen die Basissätze als Folge von Feldern mit einer maximalen Feldlänge und Typattribut beschreibbar sein. Feldendekennzeichen für verkürzte Felder ist ein Tabulator.

Der direkte Zugriff auf die Basisdatei wird über die Suchdatei ermöglicht. Die Suchdatei enthält die sortiert abgespeicherten Schlüssel mit zugeordneter Basisadresse (invertierte Datei). Es können in der Suchdatei bis zu 25 logische Suchdateien geführt werden. Eine oder mehrere logische Suchdateien sind eindeutig einer logischen Basisdatei zugeordnet. Die Suchdatei wird durch Vorgängerkompression speicherplatzsparend aufgebaut.

Weitere Systemdateien sind:

- LOG-Datei
- Protokolldatei
- Stapeleingabedatei
- Blätterdatei.

Der Anwender kann in seinen Anwenderprogrammen (Transaktionen) beliebige Anwenderdateien verarbeiten.

3. Programmsystem

KDS2000 ist streng modular aufgebaut. Wir unterscheiden folgende in sich abgeschlossene Teilkomplexe:

- Monitor
- Basisdateiverwaltung
- Suchdateiverwaltung
- Transaktionssteuerung

Der Informationsaustausch zwischen den einzelnen Moduln erfolgt über Steuerblöcke. In der 1. Ausbaustufe wird KDS2000 als Überlagerungssystem bereitgestellt. Eine Übersicht über das Zusammenwirken der Teilkomplexe gibt die folgende Darstellung:

4. Implementierung

Das größte Problem für den Anwender ist die Programmierung der eigenen Transaktionen. Es wird dabei durch das KDS-Makrosystem unterstützt. Folgende Systemdienste können über Makros angefordert werden:

- Zugriff auf die Basisdatei
- Zugriff auf die Suchdatei
- Ausgabe auf Terminal oder Protokoll
- Arbeit mit einer FCS-Anwenderdatei
- Verwaltung von Informationen im dynamischen HS-Bereich

In der 1. Ausbaustufe sind nur Makros für MAC1600 vorgesehen. Im Rahmen der Weiterentwicklung wird sich die Zahl der dem Anwender bereitgestellten nutzbarer Transaktionen für Grundfunktionen der Datenmanipulation erhöhen.

Das Generieren des Hauptprogramms erfolgt mit einem Kommandofile. In einem programmgesteuerten Dialog werden vom Anwender Informationen über seine Transaktionen und weitere Generierungsparameter abgefordert.
Mit dem so erzeugten Programm KDS.TSK kann dann mit den eingebundenen Transaktionen die Datenbasis aufgebaut und verwaltet werden.

Zum Anlegen und Pflegen des Dateisystems steht das Dienstprogramm KDSSRV.TSK zur Verfügung, das folgende Funktionen realisiert:

- Anlegen der Systemdateien
  (Basisdatei, Suchdatei, LOG-Datei, Blätterdatei)
- Definieren der logischen Basis- und Suchdateien
- Auflisten der aktuellen Dateidefinitionen für Basis- und Suchdatei
- Sichern des Datenbestandes.

5. Weiterentwicklung

Bei der Weiterentwicklung von KDS2000 wollen wir durch folgende Vorhaben die Leistungsfähigkeit und Nutzungsfreundlichkeit verbessern.

1.) Schaffung von allgemein nutzbaren Transaktionen für alle Grundfunktionen der Datenmanipulation.

2.) Erweiterung des Makrosystems und Entwicklung von Makros für komplexe Aufgaben der Datenmanipulation.

3.) Schaffung von Anschlußmöglichkeiten für Anwenderprogramme in höheren Sprachen.

4.) Herauslösen der Transaktionen aus dem Steuerprogramm als selbständige Task.

5.) Ermöglichung der parallelen Abarbeitung mehrere Transaktionen.

Werner, Klaus

Probleme des Datenschutzes in relationalen Datenbanken unter Beachtung der Schnittstelle zu statistischen Auswertungssystemen

1. Problemstellung

Moderne Informationssysteme werden in zunehmendem Maße zu notwendigen Instrumenten und Werkzeugen des wissenschaftlich-technischen Fortschritts. Daraus ergeben sich aber gleichsam eine Reihe von Aufgabenstellungen - insbesondere für Informatiker und Ökonomen - hinsichtlich des Entwurfs, der Gestaltung und der effizienten Nutzung dieser Systeme.
Eine dieser Aufgabenstellungen betrifft die Fragen des Datenschutzes. Genügte in der Vergangenheit, daß diese Fragen primär administrativ durch verwaltungstechnische Gesetzesregelungen, die sich zumeist auf die Deponie und die manuellen Zugriffs- und Auskunftsrechte bezogen, abgesichert wurden, so gelten heute gänzlich andere Prämissen.
Zweifelsfrei ist dabei die Zielstellung, der Schutz und die Sicherung der Daten und Informationen vor Verlust und unzulässiger bzw. unberechtigter Benutzung, die gleiche geblieben. Wohl aber haben sich geändert

- die Formen und Möglichkeiten der Erfassung, Abspeicherung, Änderung und Auswertung der Daten,

- die Kombinations- und Recherchemöglichkeiten von und in abgespeicherten Datenbeständen,

- die Fragestellungen der Datenintegrität unter Beachtung der Verflechtungsmöglichkeiten auf der Basis der Nutzung moderner Datenbank- und Dialogverarbeitungssysteme,

- die Möglichkeiten der Duplikat- und Teildatenmengenbildung,

- die Formen und Methoden des Zugriffs, insbesondere die Nutzung solcher Möglichkeiten, wie der Dialogverarbeitung, der Datenfernverarbeitung, der Rechnerkopplung etc.,

- die Erfordernisse und Möglichkeiten der Nachweisführung über entsprechende Datenmodifikationen und -manipulationen,

- die Geschwindigkeiten in der Datenbereitstellung und

- die Menge und Häufigkeit der anfallenden Daten und Informationen.

Diese veränderten Konstellationen bedingen notwendigerweise auch andere Betrachtungsformen des Datenschutzes. Der Begriff Datenschutz soll dabei einmal in seiner umfassenderen allgemeinen Bedeutung und zum anderen, bezogen auf die nachfolgenden Ausführungen, in einem engeren Sinne verstanden werden.
In seiner allgemeineren Bedeutung bilde der Datenschutz die Gesamtheit der Standards und gesetzlichen Regelungen zur Durchsetzung und Realisierung schutzwürdiger Belange bei der manuellen, maschinellen und automatischen Ermittlung und Bearbeitung bzw. Nutzung von Daten und Informationen.
Im engeren Sinne beinhalte er die Kontrollierbarkeit und Steuerung bzw. mögliche Protokollierung der Zugriffe zu Daten und Informationen, die in Datenbanken oder Dateien gespeichert sind und über Dialog-, Datenfern- oder herkömmliche Jobverarbeitung abgerufen oder modifiziert werden sollen, mit rechentechnischen Mitteln des Betriebs-, Datenbank- und/oder Anwendungssystems.
Aufgabenstellungen dieser Problemkreise werden integrale Bestandteile der Informationssysteme, und es gilt, die Mittel und Möglichkeiten der modernen Informationsverarbeitung zu nutzen, um auch hier entsprechende Lösungen anzubieten.
Eine zentrale Stellung in diesem Zusammenhang nehmen die Datenbanken ein. Sie stellen im weitesten Sinne eine moderne rechentechnische Realisierung der Speicherung von, der Änderung von und des Komplexzugriffs zu Datenbasen dar. Aus der Sicht der Informatik ergibt sich daraus die Aufgabenstellung, den Anwendern dieser Datenbanken Schutz- und Sicherungsmechanismen anzubieten, die o. g. Zielstellungen erfüllen.

Die Vielfalt der existierenden Datenbanksysteme unterschiedlicher Modelle und deren konkrete Implementierungen lassen dies zu einer komplexen Aufgabenstellung werden, deren Lösung eine interdisziplinäre Arbeit erfordert. Aus dem Blickwinkel des Informatikers scheint es erforderlich, zu untersuchen,

- welche konkreten Schutzmechanismen für die unterschiedlichen Datenbankmodelle bzw. -systeme existieren, erforderlich bzw. möglich sind,

- welche Schnittstellen einmal zur Betriebssystemumgebung und zum anderen zu entsprechenden Auswertungssystemen gegeben oder festzulegen sind,

- welche Zugriffsverfahren und -methoden für welche Schutzbelange geeignet und nutzbar sind und

- welcher Grad der Verallgemeinerung dieser Lösungen hinsichtlich der einzusetzenden bzw. existierenden Datenbanksysteme notwendig ist.

Im Rahmen dieses Beitrages seien einige Bemerkungen zur Problematik des Datenschutzes in relationalen Datenbanken unter besonderer Beachtung einer möglichen Schnittstelle zu einem statistischen Auswertungssystem gemacht. Die angebotenen Erkenntnisse basieren auf entsprechenden Studien und noch nicht abgeschlossenen Untersuchungen dieser Problematik im Rahmen einer vorzulegenden Dissertation zu einem gleichartigen Thema. Sie sind deshalb als ein Problembeitrag und nicht als eine Problemlösung anzusehen. Als Datenbanksystem steht MIMER (relationales Datenbanksystem der Universität UPSALA in Schweden) zur Verfügung und als entsprechende Auswertungssysteme bieten sich an PP-Statistik, SPSS, BMDP oder SAS.

2. Probleme des Datenschutzes in relationalen Datenbanken

Das relationale Datenmodell oder Relationenmodell wurde von CODD (1970 bzw. 1979) beschrieben. Es ist im wesentlichen inhaltsorientiert und grundsätzlich nicht zugriffspfadabhängig, d. h.

sowohl für das Abspeichern und Wiederauffinden von Daten ist der Inhalt der Datenstruktur von Bedeutung und keine bestimmte Zugriffsfolge. Eine Datenstruktur wird im relationalen Modell als eine Relation bezeichnet. Ein wesentlicher Gesichtspunkt bei Datenstrukturen des relationalen Modells sind die Operationen über diesen Datenstrukturen. Auch dadurch wird deutlich, daß Relationen in gewisser Weise verwandt sind mit Datenmatrizen und demzufolge für statistische Auswertungen gut geeignet scheinen.
Neben den allgemein bekannten und in den meisten Betriebssystemumgebungen realisierten Möglichkeiten der Identifikation, der Authentifikation und der Autorisierung als Standarddatenschutzalgorithmen scheinen sich insbesondere bei relationalen Datenbanken eigenständige Mechanismen einer Zugriffskontrolle anzubieten. Dabei bedingt eine derartige Zugriffskontrolle zweierlei

- ein entsprechendes Kontrollsystem und
- dazugehörige Zugriffsbeschreibungen.

In der Literatur werden Systeme, in denen man die einem Benutzer nicht zugänglichen Daten in den sogenannten Zugriffsbeschreibungen festlegt, als offene Systeme bezeichnet. Geschlossene Systeme liegen dann vor, wenn nur die dem Benutzer zugänglichen Daten beschrieben sind (need-to-know-Prinzip).
Beide Kontrollstrategien sind durch logische Operationen ineinander überführbar, so daß eine ausschließliche Festlegung auf eine der Strategien nicht erfolgen muß.
Das Kontrollsystem muß integraler Bestandteil des Datenbankbetriebssystems sein, während die entsprechenden Zugriffsbeschreibungen im Sinne der logischen Datenbeschreibungen in der konzeptu ellen Ebene getrennt erfolgen können.
Dabei bietet es sich an, diese Datenbeschreibungen ebenfalls in Form von Datenstrukturen (Relationen) darzustellen (etwa wie in MIMER v. 3). Diese Relationen widerspiegeln sinngemäß die von Prof. Dr. Wedekind theoretisch untersuchten Zugriffsmatrizen. In diesen werden funktionale Subjekt-Objekt-Beziehungen als Zugriffsbeschreibungen genutzt. Dabei gilt es insbesondere bei diesen funktionalen Subjekt-Objekt-Beziehungen zu unterscheiden zwischen

- wertunabhängigen und
- wertabhängigen Datenobjekten.

In MIMER ist es notwendig, dem Kontrollsystem wertunabhängige funktionale Subjekt-Objekt-Beziehungen anzubieten.
Prof. Dr. Wedekind schreibt, daß eine Umwandlung wertabhängiger in wertunabhängige Datenobjekte einmal durch "Manualisierung" oder zum anderen "zusätzliches Anlegen von Datenobjekten" möglich ist.
Methoden der zweiten Variante sind zum Beispiel:

- eine inhaltsabhängige Zugriffskontrolle
  Im Sinne der Operatoren über dem Relationenmodell entspräche dies einer Restriktion (oder auch where-condition bei MIMER beispielsweise), d. h. einer horizontalen Teilmengenbildung. Aus einer gegebenen oder erzeugten Relation werden definierte Tupel ausgewiesen.
  Sie beinhaltet typische Eigenschaften eines geschlossenen Systems.

- die kontextabhängige Zugriffskontrolle
  Darunter soll eine vertikale Mengen- bzw. Teilmengenbildung beispielsweise durch die Operatoren Projektion oder Join verstanden werden. Typische Eigenschaften offener Systeme sind erkennbar.
  In der praktischen Realisierung findet man häufig Kombinationen von inhalts- und kontextabhängigen Zugriffsmechanismen.

- die ablaufabhängige Zugriffskontrolle
  Ein Vorteil des relationalen Datenbanksystems ist, daß die Datenstrukturen (Relationen) relativ unabhängig voneinander existieren, sie aber sehr leicht kombinierbar miteinander sind. In analoger Weise lassen sich andererseits durch die Projektion Teile der Datenbank als neue Datenstrukturen herausfiltern.
  Die ablaufabhängige Zugriffskontrolle geht nun davon aus, daß ein Nutzer zwar auf definierte Datenstrukturen einzeln zugreift, sie aber nicht beliebig kombinieren darf.
  Weiterreichende Ausführungen dazu sind zu finden in:
  HARTSON H.H., HSIAO D.K.,
  "A Semantic Model for Data Base Protection Languages"

- die funktionale Zugriffskontrolle
  Mit dieser Art der Zugriffskontrolle beschäftigen sich unter anderem DENNING D. E., DENNING P. J., SCHWARTZ M. D. in ihrem Beitrag: "The Tracker: A Threat to Statistical Data Base Security".
  Es geht dabei um die Kontrolle des Zugriffs auf Daten, die nicht einzeln, sondern in Form von Zusammenfassungen oder von statistischen Auswertungen dem Benutzer zugänglich gemacht werden. Also das Durchschnittsalter der gerade anwesenden Damen darf man gerade noch nennen, das Alter einer einzelnen Dame sollte aber anonym bleiben.

Insbesondere die letztgenannte Zugriffskontrolle verdeutlicht Gemeinsamkeiten von Problemen des Datenschutzes in relationalen Datenbanken und zum Beispiel statistischen Auswertungssystemen.

3. Statistische Auswertungssysteme und mögliche Schnittstellen zu relationalen Datenbanken

Auswertungssysteme sind eine Untergruppe rechnergestützter Informationssysteme. HAUX definiert sie in seiner Dissertation zum Thema "Die Verwendung komplexer Datenstrukturen in statistischen Auswertungssystemen" wie folgt:

Ein Auswertungssystem A ist definiert durch:

A = (Db, Mb, Ss, DMbvs)

wobei

Db:	die Datenbank,
Mb:	die Methodenbank,
Ss:	die Menge der Schnittstellen und
DMbvs:	das Daten- und Methodenbankverwaltungssystem

von A sind.

In der gleichen Schrift definiert er ein statistisches Auswertungssystem, indem er schreibt, es sei ein Auswertungssystem gemäß o. g. Definition, welches in seiner Methodenbank - im wesentlichen - Methoden enthält, die zur Datenanalyse mit statistischen Verfahren dienen.

Es liegt auf der Hand, daß diese Begriffsbestimmung eine Zielstellung für den Aufbau eines statistischen Auswertungssystems etwa im Sinne von

- MEDUSA (Kopplung von NATURAL - einer Softwareumgebung von ADABAS - mit SAS) oder

- RAMIS (Schnittstelle zwischen dem Datenbanksystem RAMIS und dem System SAS)

darstellt, wir aber derzeit davon ausgehen können, daß Datenbank und das Spektrum der Methoden - im allgemeinen noch nicht als Methodenbank zu bezeichnen - unabhängig voneinander existieren und als mögliche Schnittstelle eine gemeinsame Datei angesehen werden kann. Diese Datei wäre nun denkbar als eine Datenstruktur im Sinne des relationalen Datenbanksystems.
Eine Zielstellung des Datenschutzes bei statistischen Auswertungen ist die Minimierung des Identifikationsrisikos, d. h. aus den im Ergebnis der Anwendung der statistischen Methoden und Verfahren resultierenden Datenstrukturen sollte es im allgemeinen nicht möglich sein, auf konkrete Subjekt-Objekt-Beziehungen zu schließen.
Hierfür werden in der Literatur eine ganze Reihe von Möglichkeiten aufgezeigt. FELLEGI I. P. nennt in seinem Bericht: "On the question of statistical confidentiality"die Methoden der Datenreduktion. Zu ihnen zählen

- die Verwendung einer Stichprobe als Datenstruktur,

- die Entfernung konventioneller Identifikatoren, wie Name, Adresse, Geburtsdaten etc., aus der Datenstruktur und

- die Anonymität der Stichprobe.

Gerade hier scheinen relationale Datenbanken gut geeignet, derartige Stichproben in Form von Relationen und damit letztlich Tabellen den Auswertungssystemen anzubieten.

Auch die von SCHLÖRER genannten zwei Formen der Outputkontrollen

- die Output-Selektion und
- die Output-Modifikation

bieten durchaus gute Ansätze, einen statistischen Auswertungsdialog mit einer entsprechenden Stichprobe in Form einer definierten Datenstruktur zwar nicht umfassend aber doch ausreichend zu sichern.
Auf dieser Ebene praktikable Lösungen in naher Zukunft zu erreichen, sollte möglich sein - den Blick in diese Richtung zu lenken, wäre eine Zielstellung dieses Aufsatzes.

Dipl.-Ing. Klaus Werner
Karl-Marx-Universität Leipzig
Organisations- und Rechenzentrum
Karl-Marx-Platz 10/11
7010 Leipzig

Zimmerling, Roland
Müller, Dietmar

LEISTAR - eine relationale Datenbanksoftware für Anwendungsprogrammierer

1. Vorbemerkungen

Zur Rationalisierung konstruktiver und technologischer Prozesse werden zunehmend automatisierte Informationssysteme (AIS) eingesetzt. Die für diese Prozesse benötigten großen Datenmengen lassen sich effektiv nur durch Datenbanksysteme verwalten. Dabei wird ein Datenbanksystem in einem AIS der technischen Produktionsvorbereitung nicht nur in direkter Kommunikation mit dem Endnutzer als Auskunftssystem (Informationsrecherchesystem) genutzt, sondern verstärkt zur Datenverwaltung in problemorientierten Programmpaketen eingesetzt. Die im zentralen Datenspeicher enthaltenen Informationen, die traditionell vom Endnutzer nur durch Informationsrecherche ausgewertet wurden, sollen also auch Problemprogrammen verfügbar gemacht werden. Für diese Nutzungsform bedarf es einer Schnittstelle, die vom Anwendungsprogrammierer in eine problemorientierte Programmiersprache eingebettet werden kann.
Hinsichtlich des Mehrebenenarchitekturprinzips von Datenbankbetriebssystemen ist diese Schnittstelle, die die problemunabhängigen Datenbankfunktionen realisiert, auf der konzeptuellen Ebene einzuordnen und soll der problemnahen Schnittstelle eines Datenbankkerns (DBK) entsprechen (vgl. ZIMMERLING 83 und MÜLLER 84).

2. Die LEISTAR-Schnittstelle

Relationale Datenbanken setzen sich auch im kommerziellen Bereich immer mehr durch. Sie werden für die verschiedensten Anwendungen genutzt und werden Bestandteil (Teilsystem) von Expertensystemen sein.
Zur Grundausstattung der 5. Rechnergeneration gehören voraussichtlich Datenbankmaschinen, die die Relationenalgebra nach CODD (vgl. CODD 71) als Schnittstelle anbieten.

Aufgrund dieser international zunehmenden Bedeutung relationaler Datenbanksysteme wurde für die LEISTAR-Operationen die Relationenalgebra zugrunde gelegt.

LEISTAR realisiert im Unterschied zu solchen Datenbanksystemen, die ebenfalls eine Schnittstelle für Anwendungsprogrammierer bereitstellen,

- alle Operationen der Relationenalgebra,
- echte Mehrtupelverarbeitung,
  (zum Vergleich - MILLER/DB realisiert nur Eintupel - also Satz-für-Satz-Verarbeitung)
- die Verarbeitung unformatierter, variabel langer Werte.

Das Ergebnis einer Zugriffsoperation (Operation der Relationenalgebra) ist eine temporäre Tabelle, die für die Dauer der Sitzung verfügbar ist.
Auf diese temporären Tabellen können wiederum Zugriffsoperationen angewandt werden.

Neben diesen Zugriffsoperationen umfaßt der Operationsumfang

- beschreibende Operationen,
- Eingabeoperationen,
- eine Ausgabeoperation,
- Operationen zum Ändern von Tabellen und Spalten,
- Serviceoperationen und
- Operationen über Datenbanken.

Die Parameter der nachstehend aufgeführten Operationen bedeuten:

DBNAM	Name einer Datenbank
DATE	Bearbeitungsdatum
TABNAM	Name einer Tabelle/Relation (bzw. Ergebnistabelle - gilt aber nur für die Zugriffs-, Ausgabe- und Serviceoperationen)
SPANAM	Name einer Spalte/eines Attributes
ALINAM	Name eines Spaltensynonyms
WERT	Ein- bzw. Ausgabefeld; Vergleichswert
VOP	Vergleichsoperator
ANZTUP	Anzahl der Tupel einer Tabelle
ERGTAB	permanente oder temporäre Ergebnistabelle
SKZ	Schlüsselkennzeichen
MASSEH	Maßeinheit
RC	Rückkehrcode

Die unterstrichenen Parameter sind Ausgabeparameter.

2.1. Zugriffsoperationen

SELECT	(TABNAM, SPANAM, VOP, WERT, <u>ANZTUP, ERGTAB, RC</u>)
RESETRI	(TABNAM, SPANA1, VOP, SPANA2, <u>ANZTUP, ERGTAB, RC</u>)
JOIN	(TABNA1, SPANA1, VOP, TABNA2, SPANA2, <u>ANZTUP, ERGTAB, RC</u>)
DIVIS	(TABNA1, SPANA1, TABNA2, SPANA2, <u>ANZTUP, ERGTAB, RC</u>)
PROJEC	(TABNAM, SPAVEK, ANZSPA, <u>ANZTUP, RC</u>)
MENGOP	(TABNA1, MOP, TABNA2, ANZTUP, <u>ERGTAB, RC</u>)

VOP = EQ, NE, GT, LT, LE, GE

MOP = IN, UN, MI

MINUS
UNION
INTERSECTION

2.2. Beschreibende Operationen

DEFTAB (TABNAM, RC)

- Aktiviert das Definieren einer Tabelle

DEFSPA (TABNAM, SPANAM, SKZ, FORMAT, LAENGE, MASSEH, RC)

- Definiert eine Spalte
- SKZ = {PS, NS}
- FORMAT = {LN, RE, CH, UN}
- LAENGE - beliebig; wenn FORMAT = "UN" dann
- LAENGE = 'VA' (variabel)

DEFALI (TABNAM, SPANAM, ALINAM, RC)

- Definiert ein Spaltensynonym

DEFEND (TABNAM, RC)

- Schließt das Definieren einer Tabelle ab
- das Definieren aller Primärschlüssel muß vor Ausführung dieser Operation abgeschlossen sein

2.3. Anzeigen der Tabellenbeschreibung

DESTAB (TABNAM, ANZTUP, ANZSPA, DATE, RC)

- Gibt Anzahl der Tupel, der Spalten und das Aktualisierungsdatum einer Tabelle aus

DESSPA (TABNAM, N, SPANAM, SKZ, FORMAT, LAENGE, MASSEH, RC)

- Gibt alle Spaltenparameter einschließlich Namen der N-ten Spalte aus

DESDB1 (ANZTAB, DATE, RC)

- Gibt die Anzahl der Tabellen und das Bearbeitungsdatum einer Datenbank aus

DESDB2 (N, TABNAM, RC)

- Gibt den Namen der N-ten Tabelle aus

DESAL1 (TABNAM, SPANAM, ANZALI, RC)

- Gibt die Anzahl der Synonyme für eine Spalte aus

DESAL2 (TABNAM, SPANAM, N, ALINAM, RC)

- Gibt den Namen des N-ten Spaltensynonyms aus

2.4. Eingabeoperationen

SETZEI (TABNAM, RC)

- Aktiviert ein neues Tupel

PUTSPA (SPANAM, WERT, RC)

- Speichert genau einen Spaltenwert in das aktivierte Tupel

2.5. Ausgabeoperationen

TABOUT (TABNAM, SPAVEK, ANZSPA, N, WERT, RC)

- Gibt die Werte der in SPAVEK spezifizierten Spalten des N-ten Tupel aus

2.6. Operation zum Ändern von Tabellen

DELTAB (TABNAM, RC)

- Löscht eine Tabelle einschließlich deren Beschreibung

DELCON (TABNAM, RC)

- Löscht nur den Inhalt einer Tabelle

DELZEI (TABNAM, ERGTAB, RC)

- Löscht die durch Recherche spezifizierten Tupel in TABNAM

ADDSPA (TABNAM, SPANAM, SKZ, FORMAT, LAENGE, MASSEH, RC)

- Fügt an die Tabelle TABNAM eine Spalte an
- SKZ = {NS}

DELSPA (TABNAM, SPANAM, RC)

- Löscht die Spalte SPANAM in der Tabelle TABNAM (alle Werte einschließlich deren Spaltenbeschreibung)

UPESPA (TABNAM, SPANAM, ERGTAB, WERT, RC)

- Ändert die Spaltenwerte der Spalte SPANAM in den durch Recherche spezifizierten Tupeln der Tabelle TABNAM

2.7. Serviceoperationen

TEMPER (ERGTAB, SPAVEK, ANZSPA, TABNAM, RC)

- Überführt die temporäre Ergebnistabelle ERGTAB in die permanente Tabelle TABNAM mit Auswahl von Spalten

SORTAB (TABNAM, SPAVEK, ANZSPA, SOFO, ERGTAB, RC)

- Sortiert eine Tabelle bzgl. mehrerer Spalten
- SOFO = {ST, FA}

DUPLI (TABNAM, SPANAM, ANZTUP, ERGTAB, RC)

- Beseitigt Duplikate bzgl. einer Spalte

2.8. Operationen über einer Datenbank

DBOPEN (DBNAM, DATE, MODUS, RC)

- Eröffnet eine Datenbank
- MODUS = {DF, NI, OU}
  - OU — vorhandene DB eröffnen
  - NI — initialisieren
  - DF — definieren

DBCLOS (DBNAM, RC)

- Schließt eine Datenbank nach Beendigung der Sitzung ab

DBCOPY (DBZIEL, DBQUEL, RC)

- Kopiert eine Datenbank

3. Die Datenbanksoftware LEISTAR

LEISTAR wurde in FORTRAN implementiert. Die Operationen können als Unterprogramme in FORTRAN-Programmen aufgerufen werden. Die LEISTAR-Unterprogramme sind dialog- und betriebssystemunabhängig und können an verschiedene Speichersysteme (DOS, OS, THS/Filesystem) angepaßt werden.

Alle Operationen müssen in DBOPEN und DBCLOS eingeschlossen sein, und beziehen sich auf eine Datenbank. Innerhalb einer Datenbank (eine Datenbank entspricht einer DA-Datei) können beliebig viele Tabellen definiert und manipuliert werden. In einer Tabelle können beliebig viele Spalten angelegt werden, die formatiert bzw. unformatiert sein können.

Die fest formatierten Spalten können vom Typ REAL, INTEGER oder CHARACTER und maximal 255 Byte lang sein. Spalten (außer Primärschlüssel) können auch unformatiert sein, d. h. deren Länge ist zum Zeitpunkt des Definierens unbekannt und wird mit 'VA' (= variabel) spezifiziert. Die Länge wird beim Manipulieren aktualisiert und kann eine maximale Länge von 30 000 Byte erreichen. Durch das Eintragen eines speziellen Operationscodes, von Position und Länge in einem COMMON-Bereich können innerhalb der unformatierten Spalte beliebige Zeichen eingefügt, angefügt, gelöscht bzw. gelesen werden. Die Werte einer solchen unformatierten Spalte können in jedem Tupel verschieden lang sein.

Die Software von LEISTAR liegt in modularer Form vor. Das heißt, vom Anwendungsprogrammierer brauchen neben einem bestimmten Grundbestand von LEISTAR-Moduln nur die Moduln, die die benötigten Operationen realisieren, zum Anwendungsprogramm dazu verbunden werden. Der Hauptspeicherbedarf hängt also vom Umfang der benötigten Moduln ab und beträgt für die Summe aller Moduln 120 K Byte.

LEISTAR ist für ESER-Anlagen in DOS und OS lauffähig und wird auf SM 4 und Bürocomputer implementiert.
Nach der Programmierung eines entsprechenden Dialoges wurde LEISTAR erfolgreich in der Lehre genutzt.
Weitere intersektionelle Anwendungen werden realisiert bzw. sind in Vorbereitung.

Literaturverzeichnis

CODD 71 Codd, E. F.
Relational Completeness of data base Sublanguages, Courant Computer Science Symposium 6, "Data Base Systems"

R. Rustin (Hrsg.) Prentice Hall, New York, 1971

MÜLLER 84 Müller, D.
Vorschlag für ein einheitliches Konzept von Datenbanksoftware für die technische Produktionsvorbereitung
TH Leipzig, Dissertation A (Entwurf), 1984

ZIMMERLING 83 Zimmerling, R., Sümisch, J.
Das DBS im AIS der TKVO
TU Dresden, Dissertation A, 1983

Anschrift:

Dr. R. Zimmerling
Dipl.-Math. D. Müller

TH Leipzig
Sektion Mathematik und
Rechentechnik
Wissenschaftsbereich 4

7030 Leipzig
Karl-Liebknecht-Straße 132

Zschockelt, Peter

Informationsorganisatorische Anforderungen an verteilte Datenbanken aus der Nutzersicht volkswirtschaftlicher Querschnittsaufgaben

1. Zum Verhältnis der Informationsorganisation zur Datenkommunikations-Software und Hardware

Ein wesensbestimmendes Merkmal der Software besteht in der Materialisierung der Organisation von Informationen und Informationsverarbeitungsprozessen. Informationen aber sind immer an die ihnen zugrunde liegenden Basissysteme/ -prozesse gebunden (vgl. auch /1/). Die damit verbundene Wechselwirkung zwischen Basissystem/ -prozeß und Information wird durch die Inhalt-Form-Dialektik bestimmt. Daraus folgen zwei grundsätzliche Aussagen:

1. Die Anforderungen der mittels automatisierter Informationsverarbeitung (AIV) zu rationalisierenden Aufgabenlösungen bestimmen wesentlich die notwendige Organisation der Informationen und damit auch die erforderliche Software. Die volkswirtschaftliche Effektivität der Softwareproduktion hängt dabei in großem Maße davon ab, inwieweit es gelingt für typische Aufgabenklassen anpassungsfähige aufgabenklassenspezifische Software (im Gegensatz zur Einzellösung einer konkreten Rationalisierungsaufgabe) bereitzustellen.

2. Die Anwendung der AIV ermöglicht ein neues qualitatives Niveau der informationellen Gewährleistung der Basissysteme/ -prozesse. Software und die zur Anwendung der Software erforderliche Informationsorganisation werden damit neben der Hardware zu einem wesentlichen Faktor der weiteren Entwicklung der Basissysteme/ -prozesse.

Für eine hypothetisch vorausgesetzte Aufgabenklasse "Rationalisierung volkswirtschaftlicher Querschnittsaufgaben" bedeutet das konkret: Die in der Literatur dargestellte Arbeitsmittelfunktion der Software erfaßt einen zweifellos wesentlichen Aspekt, reicht jedoch zur Charakterisierung der informationellen Gewährleistung komplexer Leitungssysteme nicht aus. Der Nutzen, der primär auf

die Effektivität der Arbeitsmittelfunktion der Software gegründeten Anwendung der AIV für die sozialistische Gesellschaft, besteht in der unmittelbar möglichen Steigerung des Produktivitätsniveaus. Die Nutzung technologisch ausgereifter internationaler Software (ebenso wie der Export derselben) ist dabei durchaus möglich und kann zu beträchtlichem Effektivitätszuwachs führen. Wesentlich darüber hinausgehend ist aber der strategische Charakter der Fragestellung nach dem Verhältnis zwischen Leitungs- und Informationsorganisation innerhalb des volkswirtschaftlichen Leitungssystems, weil die von der AIV ausgehenden qualitativen Einflüsse in Wechselwirkung mit dem grundlegenden Organisationsprinzip der Wirtschaftsleitung im Sozialismus, dem Prinzip des demokratischen Zentralismus, stehen. Die Software als elementare Voraussetzung jeglicher AIV realisiert unter diesem Gesichtspunkt also zugleich wesentliche materielle Bedingungen für die weitere Entwicklung des Systems der Wirtschaftsleitung. Die in den Dokumenten der SED geforderte Realisierung volkswirtschaftlich durchgängiger Anwendungslösungen der AIV (vgl. /2/), ist folglich zwar auch ein Problem der ökonomischen Wirksamkeit der Rationalisierung von Leitungsprozessen im Sinne der Effektivität von Arbeitsprozessen, jedoch in für die sozialistische Gesellschaft bedeutenderem Maße eine Grundfrage der informationellen Gewährleistung der nach dem Prinzip des demokratischen Zentralismus organisierten Leitungsprozesse durch Software, die die Fähigkeit zur Zentralisierung und Dezentralisierung von Informationen besitzt. Das aber ist genau eine der wesentlichsten Eigenschaften verteilter Datenbanken als Bestandteil lokaler und globaler Datenkommunikationssoftware (vgl. Konzept der Computer Communication Software (CCS) in /3/). Die Ausarbeitung der informationsorganisatorischen Anforderungen an verteilte Datenbanksysteme aus der Nutzersicht definierter Aufgabenklassen einerseits und die Schaffung der soft- und hardwaremäßigen Voraussetzung für den Betrieb verteilter Datenbanken sind somit sich notwendig ergänzende Bestandteile der volkswirtschaftlich effektiven Nutzung dieser zukunftsträchtigen Anwendungsform der AIV. Die frühzeitige Orientierung auf die Lösung dieser Problematik ist zugleich von erheblicher praktischer Bedeutung. So

birgt beispielsweise eine ausschließliche Dezentralisierung der informationellen Gewährleistung von Leitungsprozessen in Form autarker Anwendungslösungen auf BC-Basis (und in der Folge, die mangelnde zentrale Vergleichbarkeit und Aggregationsfähigkeit von Datenbeständen, die behinderte zentrale Zugriffsmöglichkeit zu Daten, die fehlende Integrationsfähigkeit dezentraler Datenbasen usw.) durch die starke Einbindung in die geistigen Arbeitsprozesse der Menschen die Gefahr in sich, für langfristige Zeiträume die Möglichkeiten der AIV für die sozialistische Gesellschaft nicht im erforderlichen Maße auszuschöpfen.

2. Informationsorganisatorische Grundfunktionen verteilter Datenbanken aus der Sicht volkswirtschaftlicher Querschnittsaufgaben

Die Grundanforderung an verteilte Datenbanksysteme besteht in der hohen Verfügbarkeit der in der Datenbasis des System gespeicherten Informationen. Im Gegensatz zu konventionellen Datenbanksystemen wird diese hohe Verfügbarkeit nicht primär durch die Steigerung der Zuverlässigkeit jeder einzelnen Komponente des Datenbanksystems, sondern durch die Kombination der Komponenten der verteilten Datenbank (Netzknoten und Kommunikationskanäle) angestrebt (vgl. /4/). Der Begriff der Verfügbarkeit unterliegt insbesondere hinsichtlich

- des Umfanges, der Struktur, der Systematisierungs- und Kombinatefähigkeit von Daten und
- des Zeitfaktors für die Realisierung der Verfügbarkeit

einer aufgabenklassenspezifischen Modifikation. Aus ihr leiten sich konkrete Anwenderforderungen vor allem hinsichtlich der Aufteilung des Datenbestandes auf die Netzknoten und hinsichtlich des Ausgestaltungsgrades der einzelnen Funktionsarten eines verteilten Datenbanksystems ab. Bezüglich der hohen ökonomischen Anforderungen für die Installation und den Betrieb verteilter Datenbanksysteme sind unter diesen Gesichtspunkten effektive Anwendungslösungen notwendig. Unter diesem Ziel sollen die wesentlichsten informationsorganisatorischen Besonderheiten der Aufgabenklasse "Rationalisierung volkswirtschaftlicher Querschnittsaufgaben" für folgende Datenbankfunktionen diskutiert werden:

1. Datenverarbeitungsfunktionen,
2. Zugriffsfunktionen,

3. Transportfunktionen,
4. Funktionen zur Sicherung der Datenintegrität.

Zu den Datenverarbeitungsfunktionen:

Hierunter sei besonders auf die Verteilung der Daten auf die Netzknoten verwiesen. Die Spezifik der Aufgabenklasse besteht darin, daß

- eine Datengesamtheit mit höchstem Detailliertheitsgrad an der Basis (Betrieb/Einrichtungen) existiert,
- entsprechend den Leitungsebenen eine Datenverdichtung nach den Prinzipien der Selektion, Aggregation und Kennziffernberechnung erfolgt, jedoch die Identifikation der Daten durch zusätzliche Identifikatoren notwendig wird,
- entsprechend den Leitungsebenen die Anforderungen an die terminliche Aktualität der Daten sinken (Langfristigkeit globaler Entscheidungen),
- nicht für jede Leitungsentscheidung die komplette Datengesamtheit notwendig ist (Stichprobenprinzip) und
- ein zuverlässiger Schutz der zentralisierten Datenbestände vor unkontrollierten dezentralen Zugriffen gewährleistet sein muß.

Diesen Besonderheiten entspricht eine weitestgehend primärdatenorientierte Datenbasis, die aus einer Hierarchie

a) durchgängig bis zur zentralen Leitungsebene verbindlicher Informationen (zentral notwendiger Informationsfonds),
b) für die dezentralen Netzknoten fakultativer, jedoch nach verbindlichen zentralen Regelungen organisierter Informationen (zentrale Informationsreserve) und
c) dezentral nach autarken Regelungen organisierter Informationen (dezentrale Ergänzungsinformationen)

besteht.

Die Verwaltung einer solchen Datenbasis ist durch die Führung ebenengebundener Datenkopien und deren rhythmische (z. T. nur in größeren Zeitabständen notwendige) Aktualisierung in definierten Teilmengen möglich. Dabei gilt der Grundsatz, daß die Datenaggregation und Kennziffernbildung jeweils im Netzknoten der höheren Leitungsebene erfolgen sollte, um notwendige Veränderungen

auf diesen (einen) Knoten zu lobalisieren.
Notwendige Bedingungen dieses Konzeptes sind ein für alle Netzknoten einheitliches systematisierungsfähiges Schlüsselsystem und einheitliche Definitionsvorschriften für die Erfassung und Darstellung der Informationen (Entitäten).

Zu den Zugriffsfunktionen

Entsprechend dem dargestellten Konzept sollte angestrebt werden, Zugriffe mit hoher zeitlicher Priorität (Sofortzugriffe für Recherchen und andere operative Leitungsentscheidungen) prinzipiell auf die an den einzelnen Netzknoten verfügbaren Datenkopien zu begrenzen. Der Zugriff zu den zentralen Informationsreserven bzw. zu zentral den dezentralen Datenbasen bereitgestellten Informationen kann über die Nutzung der Transportfunktionen mittels temporärer Kopien erfolgen.

Zu den Transportfunktionen

Die Transportfunktionen sind nutzbar für
- die Aktualisierung der Datenkopien auf den Netzknoten höherer Leitungsebenen in definierten Teilmengen,
- den Zugriff zu zentralen Informationsreserven und
- die physische Realisierung eines einheitlichen konsistenten Schlüsselsystems.

Die relativ geringen Anforderungen an die zeitliche Aktualität der Datenbasis (z. B. generelle Nutzung von Nachtzeiten) gestattet dabei effektive Lösungen für die Nutzung territorial und/oder zweiglich organisierter Datenkommunikationssysteme auch für volkswirtschaftliche Querschnittsaufgaben.

Zur Sicherung der Datenintegrität

Die große Dimension der für die Aufgabenklasse notwendigen Datenbestände erschwert die Wahrung der Datenintegrität beträchtlich. Sie sollte deshalb konzeptionell
- auf der Ebene der Primärdatenerfassung als komplexe Integritätskontrolle,
- auf der Ebene der Transportfunktionen als Datenprüfung beim Sender und als operationelle Integritätskontrolle beim Empfänger

organisiert werden.

3. Ein Vorschlag für das logische Datenaustauschformat

Für den Transport der Daten zwischen den Netzknoten und für die darauf aufbauende Aktualisierung der Datenkopien auf den unterschiedlichen Netzknoten empfiehlt sich ein universelles logisches Datenaustauschformat, das folgende Bedingungen erfüllt:

- Unempfindlichkeit gegen Strukturveränderungen der zu übertragenden Datenmengen,
- Unempfindlichkeit gegen Veränderungen der Datendarstellung,
- Möglichkeiten der Erweiterung/Einschränkung der über eine Teilmenge definierten Daten (Entitäten),
- Möglichkeiten zur Speicherung vor einem Nutzerzugriff geschützter Systemsteuerdaten,
- Ermöglichung einer datenunabhängigen Gestaltung der Datenverarbeitungsfunktionen durch die Speicher- und Zugriffspfadunabhängigkeit der Datenbereitstellung.

Als Grundsatzlösung bietet sich dafür das nachfolgend definierte Modell eines virtuellen logischen Datensatzes an.
Ein virtueller logischer Datensatz (VLD) wird definiert durch:

(1) Ein VLD besteht aus m logischen Sätzen der Art A.j (j=1, 2, ..., m).

(2) Jeder logische Satz der Art A.j muß definiert genau
(2a) einmal oder
(2b) 0 oder einmal oder
(2c) s-mal (s=0, 1, ..., t)
im VLD auftreten.

(3) Genau ein einmal auftretender Satz A.i ist als primärer VLD-Identifikator privilegiert.

(4) Weitere Sätze A.k≠A.i, die der Definition 2a genügen, können als sekundäre Identifikatoren auftreten.

(5) Jeder Satz der Art A.j besteht aus n Worten W.w (w=1, 2, ..., n). Die Anzahl w der Worte einer Satzart A.j ist für alle s Sätze dieser Art konstant.

(6) Jedes Wort W.w besteht aus z Zeichen Z.y (y=1, 2, ..., x). Die Zeichenanzahl z ist für jedes Wort W.w der Satzart A.j konstant.

Die Vereinigung der VLD zu einer <u>Datei</u> erfolgt über das Trägermedium des physischen Satzes (PD) gemäß der folgenden Definitionen:

(7) Ein PD ist eine um implementierungsabhängige Spezifika ergänzte Teilmenge des VLD mit variabler Zeichenanzahl, wobei gilt:

(7a) $M: \Leftrightarrow \{\text{Sätze } A.j \in VLD : \exists W.w(\text{Satz } A.j) \neq \text{leer}\}$
(Menge der real mit Informationen belegten Sätze des VLD),

(7b) $PD \sim M$ ist eine Äquivalenzrelation in der Menge VLD.

(7c) M=VLD ist möglich. M=O ist auf Grund der Definition 3 unmöglich.

(8) Eine Datei D ist die Menge aller PD eines Informationssystems (IS), die einen VLD-Typ zur Beschreibung des definierten Querschnittsproblems abbilden (z. B. Menge aller Arbeitskräftedatensätze). $D=\{PD.x \in IS : M.x \in VLD\}$

Zur Beschreibung der Anwendung von VLD seien folgende <u>Organisationsregeln</u> aufgeführt:

(9) Jedes Wort W.w dient zur Speicherung einer Entität (entity).

(10) Zu jeder Entität existiert eine Domäne (domaine), die den zulässigen Wertebereich syntaktisch und semantisch beschreibt. Sind innerhalb eines Wertebereiches nur ausgewählte Werte zulässig, erfolgt die Beschreibung in Form eines Schlüsselverzeichnisses.

(11) Die Worte W.w der gleichen Satzart A.j werden innerhalb der Datei D als gleichartige Entitäten gedeutet.

(12) Die Satzarten A.j sollten der entity-relationship-Methode entsprechend jeweils abhängige Informationsgruppen vollständig beschreiben.

(13) Die Satzarten A.j können organisatorisch als Pflicht- (nach 2a, 2c) oder Fakultativsatz (nach 2b, 2c) definiert werden.

(14) Die Beschreibung des VLD erfolgt in Form programmexterner Datenbeschreibungstabellen.

Die funktionale Einordnung des VLD in die Struktur der Computer Communication Software (vgl. /3/) ergibt sich als Schnittstelle zwischen dem CCS-Dateitransfersystem und für die Aufgabenklasse unifizierten CCS-Anwendungslösungen.

Literaturangaben

/1/ Belke, W.: Grundsätze zur Schaffung nationaler Standards auf dem Gebiet der Informationsverarbeitung, rechentechnik/datenverarbeitung 22 (1985) 1

/2/ Direktive des X. Parteitages der SED zum Fünfjahrplan für die Entwicklung der Volkswirtschaft der DDR in den Jahren 1981 - 1985, Dietz Verlag, Berlin 1981

/3/ Gollnick, D./ Heckert, R.: Heterogener Rechnerverbund - Konzept der Computer Communication Software, rechentechnik/datenverarbeitung 22 (1985) 1

/4/ Kuss, H.: Recovery in verteilten Datenbanksystemen, Dissertation an der Technischen Hochschule Karlsruhe, 1982

Verfasser: Dr. Peter Zschockelt
Berlin

SEKTION 4
Methoden und Werkzeuge zur Softwareherstellung

Bauer, Günther

Zum Einsatz von ISDOS bei der Softwareherstellung

1. ISDOS - ein Werkzeug zur Softwareherstellung

ISDOS steht für Information System Design und Optimization System-Software, also zur Unterstützung von Entwurf und zweckmäßigster Implementation von Informationssystemen. Dieses Vorhaben reicht zurück bis ins Jahr 1968, als an der Universität Michigan (USA) unter der Leitung von D. Teichroew Untersuchungen zur Realisierung begonnen wurden.

Seit 1972 sind die Komponenten PSL/PSA verfügbar. PSL - Problem Statement Language - ist eine formale Sprache zur Beschreibung von Informationsverarbeitungssystemen. Als Grundlage wurde der Systembegriff gewählt, bei dem davon ausgegangen wird, daß ein System aus Elementen mit entsprechenden Eigenschaften und Beziehungen zwischen diesen Elementen besteht. Die Einengung dieses Systembegriffs auf Informationsverarbeitungssysteme führte zu einer Menge vorgegebener Objekttypen, Eigenschafts- und Beziehungstypen. Darauf aufgebaut ist die Syntax von PSL. Der vorgesehene Anwendungsbereich ist die Spezifikation von Softwaresystemen. Verfügbar sind dafür 8 Systemaspekte, 22 Objekt- und 55 Beziehungstypen. Die Systemaspekte der Spezifikation sind (vgl. Teichroew 1977):

Systemeingabe/-ausgabe, Systemstruktur, Datenstruktur, Datenableitung, Systemgröße und -volumen, Systemdynamik, Systemeigenschaft, Projektführung.

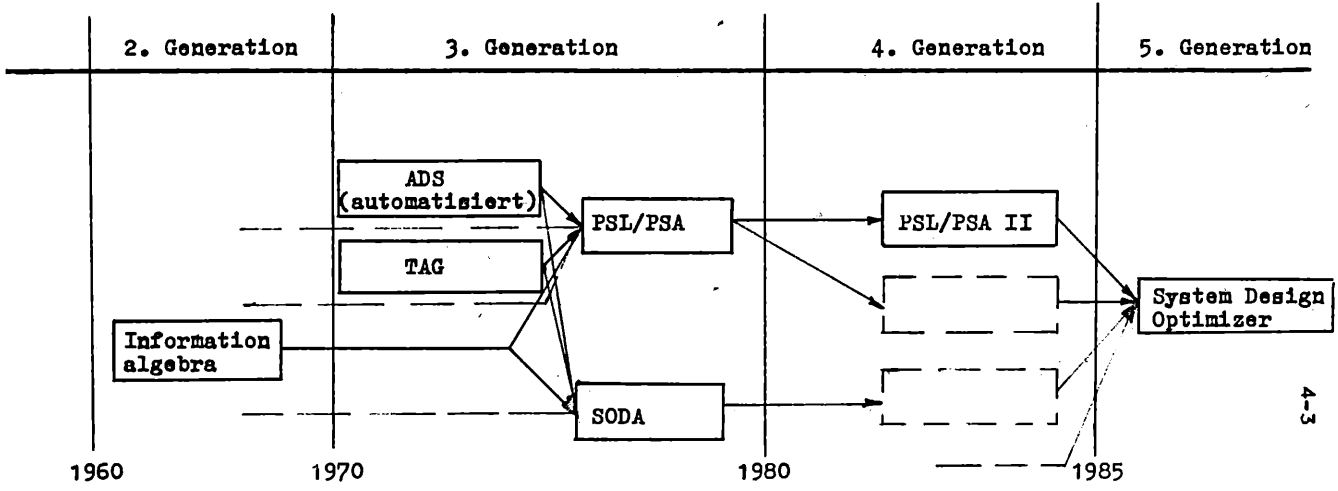
PSA - Problem Statement Analyzer - dient zur Aufzeichnung der Spezifikation in einer Datenbasis und deren umfangreiche Analyse zum Zweck der Dokumentation des vorgeschlagenen Informationsverarbeitungssystems. Die unterschiedlichen Ausgaben - hier Berichte genannt - lassen sich nach der Form und hinsichtlich des Inhalts ordnen:

Berichte

Form
- Listenformat
- Matrixform
- Grafik

Inhalt
- Datenbasismodifikation
- Referenz-Ber.
- Übersichts-Ber.
- Analyse-Ber.

Die Einordnung von PSL/PSA in die historische Entwicklung von Hilfsmitteln zur Herstellung von Software zeigt die Abbildung 1. Ein Teilthema innerhalb der Entwicklung von PSL war SODA - Syste Optimization and Design Algorithm (vgl. Couger 1982, S. 65 ff.). Es entstand dabei ein spezielles PSL, das die detailliertere Beschreibung der zu automatisierenden Prozesse ermöglicht. Mit SODA wird ein mehrstufiges Entscheidungsmodell realisiert. Dabei wird mittels SODA eine Menge von Hardwareressourcen ausgewählt, es wird die Datenorganisation für die zu verwendenden Dateien bestimmt, und es wird die Menge der erforderlichen Verarbeitungsprogramme generiert (zum prinzipiellen Aufbau von SODA und seiner Wirkungsweise s.a. Herrlich und Linder 1981, S. 143). Die historische Einordnung von SODA ist ebenfalls im Bild 1 angedeutet.
PSL/PSA wird auf höherer Stufe weiterentwickelt. Uber bereits vorhandene Nachfolgesysteme (SREM und PLEXYS) und sicher noch hinzukommende Entwicklungen wird weiterhin das Ziel verfolgt, alle Abschnitte der Softwareherstellung mit einem integrierten System rechnergestützt ausführen zu können. Diese Zielstellung wird in Abbildung 1 mit System Design Optimizer bezeichnet und stimmt mit der für ISDOS überein.



- ADS: Accurately define systems
- TAG: Time automated grid technique

Abb. 1: Entwicklung von Hilfsmitteln zur Softwareherstellung (Auszug aus Couger 1982)

2. PSL/PSA-Anwendung bei der Softwareherstellung

Ein Werkzeug ist ein "Arbeitsmittel zur unmittelbaren Bearbeitung eines Arbeitsgegenstandes (Roh-, Werkstoff, Werkstück)". Unmittelbare Bearbeitung steht hier für eine Einwirkung auf den Arbeitsgegenstand, so daß als Folge dieser eine Veränderung der Beschaffenheit des Roh- bzw. Werkstoffes oder eine Formänderung des Werkstücks eintritt. Mit dem Faktor Mittel zur Veränderung wird demzufolge das Wesentliche des Begriffs charakterisiert.
Mit Softwarewerkzeug meinen wir ein Mittel zur Bearbeitung von Software, das selbst Software ist. Arbeitsgegenstand ist somit ein Ausgangs-, Zwischen- oder Endprodukt der Softwareherstellung. Die Art des Arbeitsgegenstandes und die ausführbare Funktion sind somit bestimmende Kennzeichen für ein Softwarewerkzeug.
PSL ist, wie bereits erwähnt, ein Mittel zur Spezifikation von Softwareprodukten. Diese soll nach Möglichkeit vollständig, konsistent und eindeutig sein. Die Aufgaben von PSA sind die Analyse der eingegebenen Spezifikation und die Dokumentation jeweils geforderter Aspekte des beschriebenen Systems. Es werden dabei folgende Funktionen ausgeführt:

- Zusammenstellung umfassender Daten- und Funktionsverzeichnisse,
- Durchführung einer statischen Netzwerkanalyse zur Sicherung der Vollständigkeit der abgeleiteten Beziehungen,
- Durchführung einer dynamischen Analyse zur Anzeige zeitabhängiger Beziehungen zwischen den Daten,
- Analyse von Bedarfsspezifikationen (Systemgröße, Ressourcen).

Der Autor hatte Gelegenheit, im Rahmen einer Studie sich über die wesentlichen Funktionen von PSL/PSA (Version 2.1) zu informieren. Die dabei gewonnenen Erkenntnisse lassen sich zu folgenden Aussagen und Meinungen zusammenfassen:

Die Sprache PSL ist wirklich einfach. Für die Notation einer einzelnen Anweisung genügt eigentlich das Wissen um die Schlüsselwörter für Objekt- und Beziehungstypen und zur Konvention über die Bildung von Bezeichnern (vgl. Balzert 1981, S. 161). Die Regeln für die Zusammensetzung eines Problemstatements sind gleichermaßen einsichtig.
Die Ablage des Problemstatements in der Datenbasis und die Modifikationsmöglichkeiten ergeben sehr günstige Bedingungen für die Erarbeitung der Spezifikation. PSA akzeptiert syntaktisch richtige und mit dem bereits eingespeicherten Teil der Problembeschreibung verträgliche Erweiterungen. Dadurch wird in vorteilhafter Weise die schrittweise Entwicklung der Spezifikation unterstützt. Schrittweise kann dabei bedeuten:

- die Verfeinerung der Spezifikation von oben nach unten,
- das Sammeln von Angaben in der Reihenfolge der Gewinnung,
- die arbeitsteilige Anfertigung der Spezifikation.

Die mittels PSA erzeugbaren Berichte sind sehr vielfältig. Zur Unterstützung der Spezifikationserarbeitung sind die Zusammenstellungen in Listen- und Matrixform wie Modifikation der Datenbasis, die Referenzberichte (Formatted Problem Statement, Dictionary) und die Analyseberichte (Data-Process-Interaction) sehr gut geeignet. Letzte weisen u.a. auf Lücken im Datenfluß bzw. auf unbenutzte Datenobjekte hin.

Die grafischen Darstellungen, vorrangig zur Dokumentation entwickelt, sind nach Auffassung des Autors nicht in jedem Fall vorteilhaft. Die vorgegebene Größe der Druckerzeichen und die Blattaufteilung führen zu beträchtlichem Papierverbrauch und ziehen eine Grenze, die zu wenig Freiraum bietet. Für die bildliche Darstellung der Prozeßstruktur läßt sich bei disziplinierter Verfeinerung auch eine gute Transparenz erreichen. Bei Datenstrukturen scheint diese Form jedoch völlig unzulänglich.

Zusammenfassend ist festzustellen, daß die Funktionen von PSL/PSA gut geeignet sind, die Ausarbeitung von Spezifikationen für automatisierte Informationsverarbeitungssysteme zu unterstützen. Die Aufzeichnung der Eingaben in der Datenbasis konzentriert sämtliche Daten an einem Ort und bietet gute Möglichkeiten zur Modifikation. Die Funktionen des Umordnens und Formatierens schaffen die Voraussetzungen für die Darstellung verschiedener Aspekte des beschriebenen Systems. Damit werden die Kommunikation mit dem Auftraggeber einerseits und die Absprachen innerhalb der "Bearbeitergruppe" andererseits unterstützt. Die Prüfungen und Analysen der eingegebenen Daten in bezug auf Konsistenz, Eindeutigkeit und Vollständigkeit erhöhen spürbar die Qualität einer Systemspezifikation. Fehler, die in der Anfangsphase der Softwareherstellung verursacht und nicht erkannt werden, wirken am nachhaltigsten störend im gesamten Prozeß. Ein großer Teil dieser Fehler wird durch diese Prüfungen ausgeschlossen. Der Einsatz und die weiterführenden Arbeiten zu PSL/PSA sowie zu SODA sind als Bestätigung für die Praktikabilität des eingeschlagenen Weges anzusehen.

Literaturhinweise:

Balzert, H.: "Methoden, Sprachen und Werkzeuge zur Definition, Dokumentation und Analyse von Anforderungen an Software-Produkte" Informatik-Spektrum 4 (1981)

Couger, J.D.; Colier, M.A.; Knapp, R. W.: "Advanced System Development/Feasibility Techniques", New York 1982

Herrlich, O.; Lindner, U.: "Strukturierte Programmierung" Leipzig 1981

Dr. G. Bauer
Humboldt-Universität zu Berlin
Organisations- und Rechenzentrum
1086 Berlin, Unter den Linden 6

Kurzfassung des Vortrages für bit'85

Autor: Gunter Chmiel

Titel: Perspektiven der Entwicklung und Anwendung des Technologischen Systems zur Softwarentwicklung TESYS

Ausgehend vom gegenwärtigen Stand bei der Bereitstellung technologischer Systeme zur Softwarentwicklung und ersten Anwendererfahrungen wird in einem ersten Schwerpunkt dargelegt, welche funktionellen Erweiterungen und Kopplungskomponenten zwischen TESYS und anderen Systemen (PORG, PSU, DBS/R) in unmittelbar nächster Zeit verfügbar sein werden. Diese werden inhaltlich erläutert.

Der zweite Schwerpunkt beschäftigt sich mit der Weiterentwicklung des TESYS im Zeitraum bis 1987 und später sowie die sich daraus ergebenden Perspektiven der Anwendung, besonders unter den spezifischen Bedingungen der DDR.

Die Weiterentwicklung von TESYS besteht vor allem in der Unterstützung zusätzlicher

-Methoden (vor allem der Entwurfsphase),

-Zielsprachen und

-Zielmaschinen (Portabilität für ESER, K 1600 unter MUTOS, ACS-großes Modell)

mit dem Ziel, einen Beitrag zur Schaffung einer einheitlichen Projektierungsumgebung für Anwendungssoftware zu liefern.

Detering, Rolf; Herrlich, Ottomar; Kühle, Frank;
Lorenz, Manfred; Protzner, Sven

Entwurf für komplexe Steuerungssysteme - RESEM-Anwendung für Mikrorechner-Stellwerk

1. Einleitung

Durch den Einsatz von Computern zur Bearbeitung immer komplexer werdender Aufgaben wachsen Bedeutung und Einsatzmöglichkeiten zum rechnergestützten Soft- und Hardware - Entwurf. In der Vergangenheit wurden Unterstützungen für Entwurfsprozesse fast ausschließlich auf programmiersprachlichem Niveau angeboten. Der enorme Einfluß davor liegender, früher Phasen auf Qualität und Kosten des Finalprodukts machen aber eine Unterstützung dieser Phasen dringend notwendig. Dabei ist es wichtig, eine durchgängige Entwurfstechnologie zu schaffen, die bis hin zur Implementierung Unterstützung bietet. An eine derartige Technologie werden sehr hohe Anforderungen gestellt, die besonders eine starke Flexibilität betreffen. Operationen und Beschreibungsmittel müssen so geartet sein, daß mit ihrer Hilfe der gesamte Entwurfsprozeß unterstützt werden kann.

Bei der Betrachtung früher Phasen wird es besonders deutlich, daß eine funktionelle Aufteilung der Entwurfskomponenten auf Soft- oder Hardware noch nicht grundsätzlich vorgenommen werden kann. Daher gelten die zu zeigenden Möglichkeiten allgemein für den Entwurf von Problemlösungen; des besseren Verständnisses wegen werden sie an Software-Beispielen erläutert.

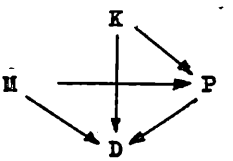
Die praktische Umsetzung der gewonnenen Erkenntnisse wird mit Hilfe der Rechnergestützten Software - Entwurfsmethode (RESEM) vorgenommen. Die Anwendung für die Praxis erfolgt für eine komplexe, sicherheitsrelevante Entwurfsaufgabe - die Steuerung eines Stellwerks.
Der Einsatz von Rechentechnik zur Steuerung von Stellwerken bedeutet einen völligen Generationswechsel von der bisher international eingesetzten Relais-Technik zum Einsatz der Mikroelektronik.

2. Zielsetzung

Die auf dem Entwurfssektor generell auftretenden Probleme bei der Beherrschung komplexer Systeme zeigen sich auf Grund der erhöhten Anforderungen verschärft für sicherheitsrelevante Anwendungen. Der zu führende Sicherheitsnachweis erfordert schon frühzeitig Festlegungen zur Systematik des Entwurfsprozesses und der dabei einzuhaltenden Qualitätskennzeichen. Diese Festlegungen müssen in Abstimmung aller am Entwurf beteiligten Personengruppen der unterschiedlichen Teilgebiete von Auftraggeber, Entwickler und Nutzer getroffen werden. Alle während der Problemlösung notwendigen Operationen müssen als integrale Bestandteile des Entwurfsprozesses aufgefaßt werden. Als wesentlichste Operationstypen betrifft das die Operationen zur

Konstruktion (K)
Modifikation (M)
Prüfung (P) und
Dokumentation (D).

Daraus können modulare Entwurfstechnologien zur Nutzung für verschiedenste Einsatzbereiche aufgebaut werden. Dabei können allgemeine Folgeabhängigkeiten zwischen den Operationstypen ausgenutzt werden.



So gilt z.B. für Modifikationsoperationen, daß sowohl die Modifikationen als auch die daraus folgenden Prüfungen dokumentiert werden müssen.

Als Gegenstand für alle Operationen im Entwurfsprozeß muß eine einheitliche Wissensbasis vorhanden sein, die zu jedem konkreten Zeitpunkt den aktuellen Zustand des Systems, also der Entwurfseinheiten und ihrer Beziehungen zueinander, widerspiegelt. Für alle Operationen gilt, daß sie für den Einsatz auch in frühen Entwurfsphasen auch die Eigenschaften der frühen Phasen berücksichtigen müssen, ohne grundsätzlich nur für diese Phasen

nutzbar zu sein. Hier sind vor allem als Charakteristika zu nennen:

- noch viele Unklarheiten
- unvollständige Beschreibungen
- inhomogener Bearbeiterkreis
- hohe Komplexität.

Daraus folgt z.B., daß Prüfungen auch dann vorgenommen werden können, wenn bestimmte Beschreibungsstufen noch nicht abgeschlossen sind. Alles vorhandene Wissen, unabhängig von seiner Abgeschlossenheit, muß schon überprüfbar sein. Damit können Prüfoperationen z.B. auch Hinweise auf zu komplettierende Stellen des Entwurfes geben.

3. Theoretische Grundlagen

Für die Durchführung eines Entwurfsprozesses ergibt sich die Notwendigkeit, Entwurfsschritte entsprechend den möglichen Operationen zu definieren. Dabei finden grundsätzliche sowie spezielle Erfahrungen des Einsatzgebietes für die Aufstellung von Problemlösungen ihren Niederschlag. Die Abstimmung der Kommunikationsschnittstelle für die am Entwurf beteiligten Personen bedingt eine hohe Flexibilität für Beschreibungsmittel und - Methoden. Daher muß der die Informationen über den Entwurfsprozeß enthaltenden Wissensbasis sehr hohe Aufmerksamkeit geschenkt werden, da sie diese Flexibilität gewährleisten muß. Auf Grund der Eigenschaften früher Phasen ergeben sich Probleme bei der Formalisierbarkeit der Beschreibungsmittel als notwendige Voraussetzung für die Rechnerstützung. Besonders für komplexe Entwurfsaufgaben wird eine standardisierte Vorgehensweise mit entsprechenden Beschreibungsmitteln angestrebt. Dieses Problem kann nicht durch feste Entwurfssprachen gelöst werden, da dabei besonders die Ausdrucksmöglichkeiten für unklare bzw. unvollständige Sachverhalte nicht gegeben wären. Dazu sind Problemlösungen als anwendungsangepaßte Modelle zu beschreiben. Das setzt Ausdrucksmittel in der Art generierbarer Fachsprachen voraus, die aber während des Entwurfsprozesses variierbar sein müssen. Durch die modellhafte Formulierung der späteren Lösung sind Untersuchungen am Modell schon lange vor der Realisierung der Lösung am Zielsystem möglich.

Da sich die Beschreibungsmittel nicht in sinnvoller Weise als feste Entwurfssprache formalisieren lassen, wurden die Definitionsmöglichkeiten für Beschreibungsmittel formalisiert. Damit ist die Anpassung an eine beliebige Sprachwelt der Nutzerumgebung durchführbar.

Bei einer solchen Vorgehensweise ergeben sich Vorteile:
- Rechnerstützung ist möglich,
- Universalität im Rahmen der Definitionsmöglichkeiten,
- hohe Variabilität/Flexibilität/Anpassbarkeit.

Für die durchgängige Unterstützung des Entwurfsprozesses muß eine Vielzahl von Problemen gelöst werden, wobei im wesentlichen Erkenntnisse aus drei Wissenschaftsdisziplinen genutzt werden können:
- Datenbanken,
- künstliche Intelligenz und
- Software-Technologie.

Für die Organisation der Wissensbasis als eines der Kernstücke eines entwurfsunterstützenden Systems sind besonders Erfahrungen aus den Bereichen Datenbanken und künstliche Intelligenz relevant. Sowohl von Seiten der Nutzerakzeptanz als auch der praktischen Realisierbarkeit besitzen relational organisierte Datenbasen große Vorteile. Daher wurde für RESEM eine Definitionsbasis auf der Grundlage eines Entity/Relationship - Ansatzes gewählt.

4. Modellansatz in RESEM

Die Bearbeitung einer Entwurfsaufgabe entspricht dann der Aufstellung eines Entity/Relationship - Modells der gewünschten Lösung. Als Entitäten treten alle Objekte des Entwurfsmodells in Erscheinung, z.B. Funktionen, Daten, Systeme, Bearbeiter, Zeiten usw. Als Relationen werden alle Beziehungen zwischen Objekten des Entwurfsmodells beschrieben, z.B. Steuerflußbeziehungen, Datenflußbeziehungen, Hierarchiebeziehungen.
Die Nutzung von Relationen kann ohne Verlust an Allgemeingültigkeit auf binäre Relationen

$$R \subseteq V \times N \quad \text{mit} \quad V \supseteq \text{Vorbereich}, \quad N \supseteq \text{Nachbereich}$$

eingeschränkt werden.

Im Entwurfsprozeß werden zwei Stufen der Modellbildung unterschieden.

4.1. Definition der Beschreibungsmittel (Rahmenmodell)

- Definition der Objekttypen OBJTYP

Sie umfassen Klassen gleichartiger Objekte > 0. Stufe, d.h.
ein OTw $\in$ OBJTYP
kann für eine Relation RELi als auch
als RELi $\subseteq$ OTw x Nj
in Erscheinung treten,
z.B. FKT für Funktionen
DAT für Daten
OBJTYP = { OT1, OT2, ..., OTw }

- Definition der Merkmalstypen MKTYP

Sie enthalten Klassen gleichartiger Objekte = 0. Stufe,
d.h. ein MTx $\in$ MKTYP
kann für eine Relation MKRj nur als
MKRj $\subseteq$ OTm x MTx
in Erscheinung treten,
z.B. Zeit für zeitliche Angaben.
MKTYP = { MT1, MT2, ..., MTx }

- Definition der Relationen mit V, N $\in$ OBJTYP

Sie beschreiben Beziehungen zwischen Objekten > 0. Stufe, d.h.
solche Objekte, die wiederum beschrieben werden können.
RELi $\subseteq$ OTk x OTp
z.B. Hierarchiebeziehung
BESTEHT_AUS $\subseteq$ FKT x FKT
RELATION = { REL1, REL2, ..., RELy }

- Definition der Relationen mit V $\in$ OBJTYP und
N $\in$ MKTYP

Sie ordnen Objekten > 0. Stufe Merkmalsgrößen zu, d.h.
Objekte = 0. Stufe.
MKRj $\subseteq$ OTr x MTs
z.B. ABSCHLUSSTERMIN $\subseteq$ FKT x ZEIT
MKREL = { MKR1, MKR2, ..., MKRz }

Die Steuerflußbeschreibung über Relationen wird für komplexe Entscheidungssituationen sehr aufwendig und unübersichtlich,

daher werden hier Entscheidungstabellen eingesetzt. Durch die Einschränkung auf Tabellen des einfachen Typs mit Eintreffer - Eigenschaft und der ausschließlichen Angabe von Ende (E) und nächster Takt (n.T.) in der Verbundaktion wird generell ein wohlstrukturierter Steuerfluß gewährleistet. Außerdem werden damit gute Analysevoraussetzungen bzgl. Vollständigkeit, Widerspruchsfreiheit und Redundanz geschaffen. Im Aktionsteil können zur Angabe von Sequenzen bzw. Parallelitäten Ordinalzahlen je Regel angegeben werden; gleiches gilt für Wertezuweisungsaktionen im Informationsteil.

Die methodischen Erkenntnisse aus der Softwaretechnologie lassen sich dann durch Vorgabe eines bestimmten Beschreibungsumfanges des Rahmenmodells umsetzen. Dieser Beschreibungsumfang kann entsprechend der zugrunde gelegten Methodik des Entwurfsprozesses schrittweise erweitert bzw. modifiziert werden. Mit Hilfe der Grundoperationen und der Beschreibungsmittel des Entwurfsmodells lassen sich Schrittfolgen festlegen, die der Umsetzung einer Entwurfstechnologie entsprechen.

4.2. Nutzung der Beschreibungsmittel (Nutzermodell)

Hier wird das konkrete Nutzerproblem beschrieben, in dem die Entitäts-Tupel entsprechend den definierten Relationen angegeben werden.

z.B. FKT A BESTEHT_AUS A1, A2

bedeutet, daß die Relation BESTEHT_AUS um die Tupel (A, A1), (A, A2) erweitert wird.

Außer den Konstruktionsoperationen für Rahmen- und Nutzermodell sollen noch Prüfoperationen vorgestellt werden.

Allgemein müssen diese es erlauben, logische Ausdrücke mit Bezugnahme auf die Wissensbasis des Entwurfssystems zu formulieren. In RESEM werden prädikatenlogische Ausdrücke der 1. Stufe mit den logischen Operatoren UND, ODER, NICHT, IMPL, AEQU und den Quantoren EXIST, FORALL verwendet. Die Bezugnahme auf die RESEM - Wissensbasis wird über das Standardprädikat ASSO (d.h. Assoziation zum Wissen) realisiert, das über gleiche Angabemöglichkeiten wie bei Konstruktionsoperationen verfügt. Mit Hilfe dieser Ausdrücke können Aussageform - Verbindungen (GILT) oder Kennzeichnungsform - Verbindungen (SUCHE * ALLE) gebildet wer-

den, in denen Variable der Grundbereiche OBJTYP, MKTYP, RELATION, MKREL sowie Objekte auftreten können.

Z.B. (GILT (OBJEKT OB1) als Variable
(UND (ASSO FKT OB1 BESTEHT_AUS A1)
(ASSO FKT OB1 BESTEHT_AUS A2)))

Ergebnis: true

5. Entwurfsschritte

Besonders deutlich tritt die Notwendigkeit einer durchgängigen Entwurfsmethode auf dem Gebiet der sicherheitsrelevanten Prozeß - Steuerung zu Tage. Hier wird RESEM in der Praxis zur Modellierung der Steuerung eines Stellwerks auf Rechner - Basis eingesetzt. Die angeführten Entwurfsschritte stellen somit auch eine gewisse Rückkopplung der Erfahrungen bei der Bearbeitung der Entwurfsaufgabe dar.

Die Modellarbeit wurde als Kernstück in folgenden Gesamtentwurfsablauf eingebettet:

S.1. Zielfestlegung, Aufgabenstellung
S.2. Analyse und Modellbildung des Basissystems
S.3. Analyse des gewünschten Systemverhaltens; Synthese und Überprüfung des Steuersystems
S.3.1. Definition des Rahmenmodells
S.3.2. Definition des Nutzermodells
S.3.2.1. Entwurf durch schrittweise Verfeinerung
S.3.2.2. Überprüfung ohne Echtzeiteigenschaften
S.3.3. Ableitung und Untersuchung des Steuernetzes
S.3.4. Echtzeituntersuchungen
S.4. Vollständige Algorithmierung, Realisierungskonzeption
S.5. Kodierung für Zielrechner
S.6. Testung der Realisierung
S.7. Installation, Inbetriebnahme
S.8. Wartung und Pflege

Es folgen einige ausgewählte Bemerkungen zu typischen Entwurfsschritten.

S.1. Zielfestlegung

1. Charakteristika für das Mikrorechner-Stellwerk:
- Umfangreiche Problematik (für größeren Bahnhof bisher ca. 25000 Relais mit 150000 Kontakten)

- Hohe Sicherheitsverantwortung (weniger als eine gefährdete Signalzugfahrt auf 10^{10} Signalzugfahrten zugelassen, deshalb auch hochgradig fehlerfreie Software nötig)
- Volkswirtschaftlich bedeutsame Aufgabe
- Lange Lebensdauer (normative Nutzungsdauer ca. 30 Jahre; kein "Wegwerfprojekt", Generationswechsel bei Rechentechnik einkalkulieren).

2. Das zugehörige Basissystem, d.h. das Steuerobjekt, enthält als Außenanlagen z.B. folgende Basiselemente (Zahlen für größeren Bahnhof): 150 Weichen, 180 Signale, 150 Isolierabschnitte, 15 Streckenblock-Gleise, 3 Wegübergangssicherungsanlagen usw. Da die Außenanlagen von bisheriger Technik übernommen werden, wird im folgenden der Entwicklungsschritt S.2. nicht betrachtet.

S.3. Analyse des gewünschten Systemverhaltens, Synthese und Überprüfung des Steuersystems

S.3.1. Definition des nutzerspezifischen Rahmenmodells

Es sind die Beschreibungselemente (Objekttypen, Relationen) festzulegen. Für das Mikrorechner-Stellwerk wurden definiert (Ausschnitt):

OBJTYP = { FKT, DAT ... }

RELATION = { BESTEHT_AUS, ..., SENDET }

BESTEHT_AUS $\subseteq$ FKT x FKT $\cup$
DAT x DAT

SENDET $\subseteq$ FKT x DAT

usw.

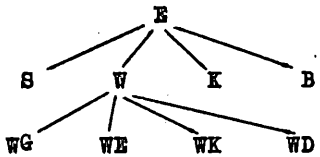
S.3.2. Entwicklung des Nutzermodells

S.3.2.1. Entwurf durch schrittweise Verfeinerung und funktionelle Dokumentation

a) Methodik des Entwurfes

Je Verfeinerungsschritt sind mindestens folgende sechs Teilschritte durchzuführen; die ersten drei betreffen die Funktionen (Aktionen), weitere drei betreffen die logischen Daten (Informationen).

1. Hierarchie für Funktionen bilden



Die Graphik zeigt eine Stelle der zu entwerfenden Funktionen als Hierarchie-Struktur: Die zugehörige RESEM-Sprachform (für die Eingabe in den Rechner) lautet bei der Beschreibung der Funktion Weiche W:

FKT W ENTHALTEN_IN E
FKT W BESTEHT_AUS WE, WK, WD, WG

Die erste Zeile dieser Sprachform drückt die Einordnung in die nächst höhere Hierarchieebene aus, die zweite Zeile beschreibt die Untergliederung in die nächst niedere Ebene.

2. Problembedingten Steuerfluß beschreiben

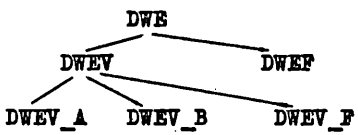
Das heißt: Wie sind die zu einer Untergliederung gehörenden Funktionen miteinander verknüpft? Das Beschreiben geschieht in Entscheidungstabellen. Sie sind bei RESEM so reglementiert, daß Wohlstrukturiertheit entsteht: Zugelassen sind nur die Verknüpfungen Sequenz, Alternative (einschließlich Case-Struktur), Zyklus und Parallelität, ferner Unabhängigkeit. Als Verbundaktion Va sind nur zugelassen Ende (E) und nächster Takt (NT), kein GOTO. Es müssen Eintreffer-Entscheidungstabellen sein: Zu jedem Zeitpunkt trifft nur eine Spalte zu (außer ELSE). Ein Beispiel folgt am Ende des Aufsatzes.

3. Eventuell nutzerspezifische Angaben für Funktionen notieren

Zum Beispiel Synchronisationsrelationen, Zeitforderungen oder Zuordnung zu den verschiedenen Teilsystemen durch die Relation GEHOERT_ZU.

4. Hierarchie für Daten bilden

Das geschieht analog zu der Hierarchiebildung für Funktionen.



RESEM-Sprachform:

DAT DWEV ENTHALTEN_IN DWE

DAT DWEV BESTEHT_AUS DWEV_A, DWEV_B, DWEV_F

5. Datenfluß beschreiben

Zum Beispiel lautet der Datenfluß in RESEM-Sprachform aus der Sicht der Funktion WEF:

FKT WEF EMPFAENGT DWEV_A, DWEV_B, DWEV_F

FKT WEF SENDET DWEV_F.

An anderer Stelle wird der Datenfluß aus der Sicht der Daten beschrieben, um alle Angaben doppelt zu haben - als Umkehr-Relation. Das entspricht den in der Sicherungstechnik gern benutzten Verdopplungen aus Sicherheitsgründen.

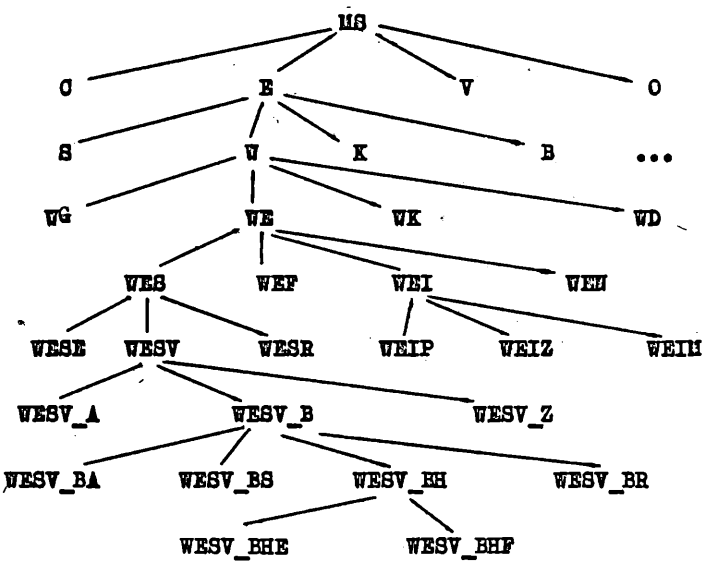
6. Eventuell nutzerspezifische Angaben für Daten nennen

Zum Beispiel Zuordnung eines Datums zu verschiedenen Teilsystemen durch die Relationen GEBILDET_IN und BENOETIGT_VON.

Entsprechend diesen sechs Teilschritten vollzieht sich der Entwurf für jedes Problemsegment von der obersten bis zur untersten Ebene. Dabei können viele Entwurfsbearbeiter parallel arbeiten.

b) Übersicht über mehrere Ebenen der Hierarchie der Funktionen

Mikrorechner-Stellwerk



Die Graphik zeigt in Baumform acht Hierarchieebenen. Dabei wurde

stets nur ein Zweig weiter verfeinert; auch die übrigen Zweige besitzen (nicht dargestellt) weitere Untergliederungen.
Die sinnvolle Hierarchiebildung ist ein schöpferischer Vorgang mit Freiheitsgraden. Es wurden schon verschiedene Varianten der Hierarchiebildung untersucht. Hat man eine Hierarchie entworfen, so ergibt sich der Steuer- und Datenfluß dann meist folgerichtig entsprechend den stellwerkstypischen Aufgaben.

c) Detailbeispiel für ein Funktionssegment einer unteren Hierarchieebene (siehe nächste Seite)

Nach der Überschrift folgen in RESEM-Sprachform die relationalen Angaben
- zur Hierarchie (ENTHALTEN_IN und BESTEHT_AUS)
- zum Datenfluß (EMPFAENGT und SENDET; die Kurzform aller Daten beginnt mit D).

Voran steht jeweils die Kurzform, danach die Langform.

Weiter unten ist die Entscheidungstabelle auszugsweise dargestellt:
Im Aktionsteil ist neu, daß je Regel Sequenzen durch fortlaufende, Parallelitäten durch gleiche Ordnungszahlen ausgedrückt werden.
Den letzten Teil der Entscheidungstabelle bildet der Informationsteil. Dieser wurde neu aufgenommen, um mögliche Wertzuweisungen für Variable ausdrücken zu können.

Insgesamt beinhaltet RESEM sowohl die Methode des Vorgehens (6 Teilschritte gemäß a)) als auch die Bereitstellung der Beschreibungsmittel (RESEM-Sprachform und -Entscheidungstabellen).

8.3.2.2. Überprüfung ohne Echtzeiteigenschaften

Nachdem - wie beschrieben - der Entwurf durch den Menschen erfolgte, schließen sich rechnergestützte Überprüfungen an. Dazu gehören:

- Prüfung auf definierte Syntax gemäß Rahmenmodell.
- Prüfung auf Logik des Entwurfes gemäß beschriebenen Relationen auf Widerspruchsfreiheit, Vollständigkeit usw.

  Beispiel:

```
(SUCHE*ALLE 10 (OBJEKT OB OBJTYP OT)
  (NICHT
    (EXIST (OBJEKT OB1)
```

Forts. übernächste Seite

	WESV_BHEH	FKT_WESV_BHE_HAUPTFAHRTELEMENT
ENTHALTEN_IN:	WESV_BHE	FKT_WESV_B_EINZELHILFSAUFLOESUNG
BESTEHT_AUS :	OEDZ_2	FKT_OEDZ_2_MINUTEN
	OMI	FKT_OM_INFORMTEIL_VERARBEITUNG
	WESV_BHEH_D	FKT_WESV_BHEH_DV_AUFLOESEN
	WESV_BHEH_Q	FKT_WESV_BHEH_QV_AUFLOESEN
	WESV_BHEH_A	FKT_WESV_BHEH_MITAUFLOESUNG
	OF	FKT_O_FEHLERBEHANDLUNG
EMPFAENGT :	DEZV_E	DEZV_ELEMENTEZEIGER
	DWEV_FVFI	DWEV_FVF_SPITZ
	DWEV_FVFL	DWEV_FVF_LINKS
	DWEV_FAQS	DWEV_FAQ_SPITZE
	...	
SENDET :	DWEV_FVFI	DWEV_FVF_SPITZ
	DWEV_FVFL	DWEV_FVF_LINKS
	...	

WESV_BHEH	A	B	C ...	ELSE
DEZV_E . DWEV_FVFI = 'VERSCHLOSSEN	L	L	L	-
DEZV_E . DWEV_FVFL = 'VERSCHLOSSEN	O	L	O	-
DEZV_E . DWEV_FAQS = 'ANGEREGT	L	O	O	-
...				
OEDZ_2	1	1	1	-
OMI	2	2	2	-
WESV_BHEH_D	-	-	2	-
WESV_BHEH_Q	2	-	-	-
WESV_BHEH_A	3	3	3	-
OF	-	-	-	1
VA	E	E	E	E
DEZV_E . DWEV_FVFI := 'UNVERSCHLOSSEN	1	1	1	-
DEZV_E . DWEV_FVFL := 'UNVERSCHLOSSEN	-	1	-	-
...				

(ODER (ASSO OT OB BESTEHT_AUS OB1)
(ASSO OT OB SENDET OB1)))))

Es sollen (max. 10) Objekte mit den zugehörigen Objekttypen gesucht werden, für die weder Hierarchiebeziehungen entsprechend BESTEHT_AUS noch Datenflußbeziehungen entspr. SENDET angegeben wurden. Damit können also Stellen im Entwurf gefunden werden, die bezüglich dieser Beziehungen unvollständig sind und noch weiter bearbeitet werden müssen; dies würde z.B. festgestellt für FKT WE.

- Funktionelle Ablauftests
  (Überprüfung auf funktionelle Richtigkeit,
  Testung des Zusammenspiels zwischen Steuer- und Basissystem vor Programmierung/Implementierung).

Spätere Entwurfsschritte (ab S.3.3.) enthalten Netzuntersuchungen, Echtzeituntersuchungen usw.

Entwicklungsstand

RESEM wird in der Praxis für das vorgestellte Entwurfsbeispiel angewendet. Die Rechnerstützung der Entwurfsmethode RESEM ist implementiert als Experimentalsystem auf Basis der Sprache LISP (TULISP für ESER).

Die Betreuung der RESEM-Anwendung erfolgt durch eine Systementwicklungsgruppe - sowohl bezüglich der Beschreibungsmittel als auch der Entwurfsmethodik. Dazu werden Nutzerhinweise erarbeitet und Seminare durchgeführt.

Literatur:

Balzert, H.:
Die Entwicklung von Software-Systemen
Reihe Informatik, Band 34, Bibliographisches Institut,
Mannheim, Wien, Zürich 1982

Detering, R., Bandke, G., Lorenz, M.:
Software-Entwurfsmethode RESEM -
eine Unterstützung der frühen Entwurfsphasen
WBZ MKR/IV Heft 71/84

Fickas, St.:
The Mechanisation and Documentation of Software Specification
University of Oregon, May 1984

Verfasser:

Dr.-Ing. Rolf Detering, Prof. Dr.-Ing. habil. Ottomar Herrlich,
Frank Kühle;
Technische Universität, Sektion Informationsverarbeitung
8027 Dresden, Mommsenstr. 13

HD Dr.-Ing. Manfred Lorenz, Dipl.-Ing. Sven Protzner:
Hochschule für Verkehrswesen, LFG Verkehrssicherungstechnik,
8072 Dresden, PSF 103

Dipl.-Math. P.Döring / Dr. A.T.Lassig

Unterstützung von Modellentwicklungsarbeiten durch das Dialog-Programmsystem MOVAS

1. Einführende Bemerkungen

Die Entwicklung von Modellen erfolgt in mehreren Arbeitsstufen. Dabei treten neben Problemen, die eine theoretische Arbeitsweise erfordern, auch solche auf, die mit einem mehr oder weniger großen numerischen Aufwand verbunden sind. Das betrifft häufig Optimierungsrechnungen (zum Zweck der Parameteranpassung). Daneben ergibt sich die Notwendigkeit, mit dem Modell eine Reihe von Simulationsexperimenten durchzuführen, deren Ergebnisse anschließend ausgewertet werden müssen. Dabei kann es sich beispielsweise um Monte-Carlo-Simulationen handeln. Eine andere Aufgabe wäre die Untersuchung der Parametersensitivität des Modells, d.h. die Feststellung, welchen Einfluß die einzelnen Modellparameter auf die Modellergebnisse haben. Häufig erweist sich auch eine umfangreiche Datenanalyse als notwendig. In allen Fällen ergibt sich meist ein beträchtlicher numerischer Aufwand. Es müssen deshalb Computer zu Hilfe genommen werden. Dabei ist es günstig, die Arbeit mit dem Modell und die Auswertung der Ergebnisse variabel zu gestalten, da es häufig erst nach einer Teilauswertung möglich wird, Entscheidungen über weitere Arbeitsschritte zu fällen. Deshalb erweist sich ein allgemeiner, dialogfähiger und modellunabhängiger Testrahmen für Modellvalidierungsarbeiten als ein gutes Hilfsmittel.

Ein diesem Ziel entsprechendes Konzept wurde mit dem Programmsystem MOVAS entworfen. Anhand der Teile SENSIT (Sensitivitätsanalyse) und MODAW (Auswertealgorithmen) sollen einige Aspekte dieses Systems vorgestellt werden.

2. Allgemeine Charakteristik des Systems

Das MOVAS-System besteht aus einer Reihe von FORTRAN-Programmen. Es arbeitet im Dialog unter TSO-Bedingungen.

Dabei werden berechnete Ergebnisse stets auf dem Bildschirm angezeigt. Der Nutzer kann dann entscheiden, ob sie zusätzlich gedruckt bzw. in eine Ergebnisdatei ausgegeben werden sollen.

MOVAS ist nicht an ein bestimmtes Modellkonzept gebunden. Es sind lediglich eine Reihe von Koppelbedingungen zwischen den das Modell repräsentierenden Programmen und den Verarbeitungsprogrammen zu erfüllen. Die Kopplung wird dann mit Hilfe einiger normierter Unterprogramme vorgenommen. Diese Programme realisieren die modellspezifischen Eingabeoperationen, das Ausführen der Modellrechnungen, das Sichern und Wiederherstellen des Modellausgangszustands sowie Bezugnahmen zwischen Modellgrößen und Verarbeitungsprogrammen. Die Parametervermittlung erfolgt dabei mit Hilfe von COMMON-Blöcken.

Die Erstellung der Koppelprogramme wird durch ein zum System gehöriges Hilfsprogramm unterstützt. Es konstruiert mit Hilfe eines Dialogs mit dem Nutzer einen FORTRAN-Quelltext, der dem Standard der Schnittstellen entspricht.

Die Bezugnahmen auf die Modellgrößen erfolgt während der Arbeit mit dem System durch vom Nutzer selbst festzulegende Bezeichnungen.

3. Arbeitsweise des Teilsystems SENSIT

Für die Sensitivitätsanalyse eines Modells werden die Modellergebnisse, berechnet für verschiedene variierte Parameterwerte, benutzt. Aus diesen Ergebnissen werden weitergehende Kennwerte hergeleitet, die dann durch MODAW untersucht werden können. Diese Kennwerte repräsentieren den Verlauf der Modelltrajektorie für die verschiedenen Parameterwerte, die Ableitung der Modelltrajektorie nach den Parametern des Modells, die relativen Änderungen der Modelltrajektorie bei Parametervariationen und das Verhältnis zwischen den relativen Änderungen der Modelltrajektorie und der Modellparameter.

Es gibt 2 generelle Möglichkeiten, die Ergebnisse der Sensitivitätsanalyse auszuwerten. Die erste Möglichkeit besteht nach der Erzeugung der Modellergebnisse. In diesem

Fall kann aber nur auf die Ergebnisse einer Parametervariation, nämlich der zuletzt durchgeführten, zurückgegriffen werden. Das wird i.a. nur für eine erste Auswertung hinreichend sein. Günstiger ist es, die berechneten Ergebnisse in Zwischendateien auszugeben. Mit Hilfe eines speziellen Programms des MOVAS-Systems können diese Dateien dann beliebig gemischt werden. Dabei können dann die Modellergebnisse verschiedenster Parametervariationen ausgewertet werden.

4. Das Teilsystem MODAW

Die bereitgestellten Modellergebnisse werden an Hand von Parameterfeldern an MODAW übergeben. Die Vermittlung von Strukturparametern für dieses Teilsystem erfolgt durch COMMON-Blöcke. Zur Auswertung von Modellergebnisgrößen können folgende statistische Prozeduren gewählt werden :

- Tabellierung
- Grafische Darstellung
- Berechnung statistischer Maßzahlen
- Berechnung der Matrix der Korrelations- und Autokorrelationskoeffizienten mit beliebiger Verschiebung
- Klasseneinteilung, Häufigkeitsanalyse mit Histogrammausgabe
- Chi-Quadrat-Anpassungstets auf Normal-,Exponential-,Gleich-, Poisson- und Binomialverteilung
- Ausreißertest

Vor der Durchführung dieser Prozeduren können die Schrittweite, der Zahlenbereich und der Umfang beliebig festgelegt werden, mit denen dann die Auswertung erfolgt. Für mit Datum versehene Modellergebnisse kann die Auswertung nach Monaten und Dekaden aufgeschlüsselt erfolgen. Das System arbeitet in mehreren Zyklen, die die Wiederholung einer Prozedur oder eines Teilsystems ermöglichen. Auf Abb. 1 ist die Zyklusarbeit von MODAW schematisch dargestellt.

Dipl.-Math. Peter Döring
Dr. rer. nat. An Thu Lassig
AdL der DDR
Inst. f. Landwirtsch. Information
und Dokumentation
1086 Berlin , PSF 1295

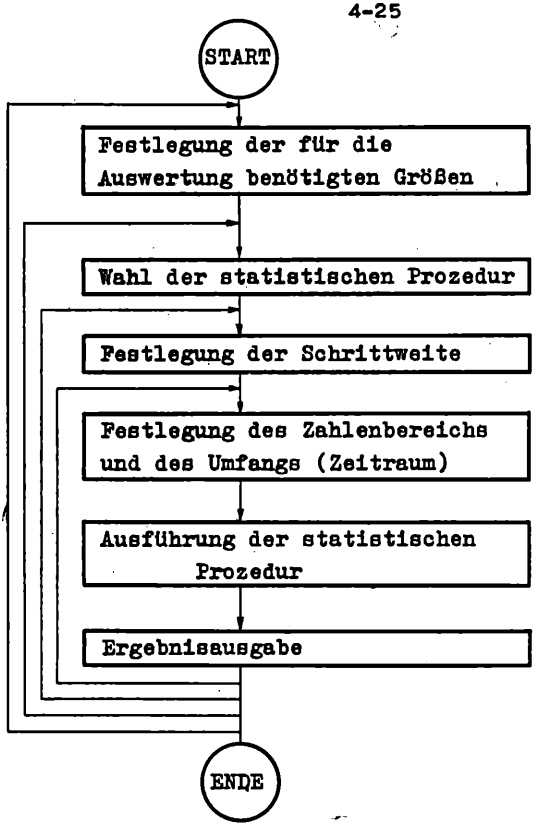


Abb. 1 : Zyklusarbeit von MODAW

Fischer, J.

Untersuchungen zur Simulation von Concurrent Pascal und Modula-Programmen

1. Einführung

Die Erstellung von Simulationsmodellen komplizierter Rechnersysteme (Rechnernetze, verteilte Datenbanken, Multiprozessorsysteme u.a.) sowie das computergestützte Experimentieren mit diesen Modellen hat sich als eine nützliche Methode hinsichtlich der Gewinnung von Aussagen zu Verhaltensverhersage, Leistungsvermögen, Leistungsverbesserung und Zuverlässigkeit dieser Systeme erwiesen. Die Attraktivität dieser Methode wächst in dem Maße wie es gelingt, sie bereits in den Entwurfsprozeß der Rechnersysteme (d.h. sowohl die Konzipierung der Hardwarekonfiguration als auch bei der Entwicklung der benötigten Software) mit einzubeziehen /1/. Die Verwendung von leistungsfähigen höheren Programmiersprachen mit speziellen Sprachkonstrukten zur Separierung unmittelbar maschinenabhängiger Teile (low-level-facilities) erweist sich dabei sowohl für einen rationellen und sicheren Entwurf der Software als auch für deren Analyse als vorteilhaft. Concurrent Pascal/2/ und Modula/3/ sind solche Sprachen, die in der DDR inzwischen auch verfügbar geworden sind.

Grundlage für die Verhaltensanalyse eines Rechnersystems mittels Computersimulation ist die Erstellung eines Modells (als Einheit von Hardware-, Software- und Arbeitslastmodell), das die wesentlichen Eigenschaften des Originals hinsichtlich eines festgelegten Untersuchungszieles widerspiegeln muß. Der Abstraktionsgrad des Modells wird dabei maßgeblich

- von der Zielstellung der Simulationsstudie,
- von der Komplexität des zu untersuchenden Systems,
- von den rechentechnischen Möglichkeiten (Speicherplatz, Rechenzeit) zur Durchführung der Experimente am Modell
- und vom Entwurfs- und Untersuchungsstadium

bestimmt. Je abstrakter nun dieses Simulationsmodell ist, um so größer ist jedoch der Aufwand für dessen Validierung. Die Validierung erfordert den Nachweis der Äquivalenz von modelliertem und realem Verhalten des Rechnersystems bei Variation der Ar-

beitslast und festgelegter Bewertung. Dies ist meist der aufwendigste Teil der Simulation.

Der hier vorgestellte Ansatz verfolgt das Ziel, Programmierwerkzeuge bereitzustellen, die es gestatten, auf einem leistungsfähigem Rechner A die Softwareentwicklung für ein Rechnersystem B (Multiprozessorsystem, Echtzeitsteuerungssystem, lokales Netz) vorzubereiten, ehe B zur Verfügung steht. D.h., auf A wird zunächst ein abstraktes Modell von B softwaremäßig zu realisieren sein und danach kann die Modellsoftwareentwicklung beginnen. Eine wesentliche Bedingung für die Verwendung dieser Methode als entwurfsunterstützendes und entwurfsbegleitendes Werkzeug besteht in erster Linie darin, daß sich Modell- und Originalsoftware ohne Schwierigkeiten in einander überführen lassen /4/. Ideal wäre es sicher, wenn die Modell- und die Zielimplementationssprache zusammenfallen. Somit stellt sich die Frage nicht nur nach den Einsatzmöglichkeietn moderner Systemimplementationssprachen schlechthin für die Simulation, sondern auch danach, ob es gelingt, daß die Modellsoftware strukturäquivalent zur Orginalsoftware entwickelt werden kann. Dies ist keine triviale Angelegenheit, da z.B. weder Modula noch Ada eine Instanzenbildung von Modulen bzw. von Paketen zulassen, was sofort zum Problem wird, wenn z.B. ein Modell eines verteilten Systems zu erstellen ist.

2. HSMOD - ein Werkzeug zur Modellierung von Hard- und Software

Da es sich gezeigt hat, daß das Klassenkonzept von Simula/5/ mächtig genug ist, um als Modell für solche Sprachstrukturen wie CLASS, MONITOR und PROCESS von Concurrent Pascal oder wie MODULE und PROCESS von Modula zu fungieren und darüber hinaus Simula als höhere Programmiersprache effektive Mittel von Hause aus für die Simulation bereitstellt, wurde hier dazu übergegangen, ein Werkzeug für die Modellierung von Rechnersystemen in Simula bereitzustellen. Hinzu kommt, daß hier keine Strukturverletzungen beim Übergang Modell-Original auf Grund der Möglichkeit der Instanzenbildung von abstrakten Datenstrukturen zu erwarten waren.

HSMOD entstand als Spracherweiterung von Simula (Unterklassenbildung von SIMULATION), wozu das System OASIS/6/ wesentliche Anregungen vermittelte. HSMOD unterstützt die Konstruktion von

abstraketn Geräten, Prozessoren und Übertragungskanälen, in dem es dafür vordefinierte Bausteine bereitstellt. Ebenso werden auch Bausteine zur Konstruktion der Modellsoftware bereitgestellt, wobei aber noch keine zielsprachtypischen Hilfsmittel bereitgestellt werden. Dies ist weiteren HSMOD- Erweiterungen vorbehalten. Die als abstrakte Datentypen umgesetzten Bausteine von HSMOD sind mit einigen ihrer wichtigsten Attributprozeduren in Abb.1 dargestellt. Die Pfeile deuten die Typerweiterungen (Unterklassenbildung) an.

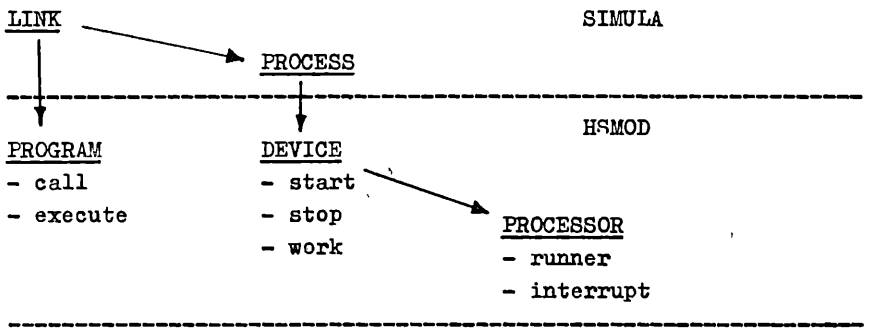


Abb. 1 Bausteine von HSMOD

Die Attribute von PROGRAM und PROCESSOR wurden so entworfen, daß ein PROGRAM-Objekt nur dann abgearbeitet wird, wenn es der augenblickliche "runner" eines PROCESSOR-Objektes ist. Die "call"-Funktion wird innerhalb eines Modellprogramms dazu benutzt, um ein anderes PROGRAM-Objekt aufzurufen. Mit der "interrupt"-Funktion ist es möglich, die Abarbeitung des laufenden Programms eines PROCESSOR-Objektes zu unterbrechen und eine entsprechende Behandlungsroutine zu aktivieren. Mit "execute" läßt sich der Verbrauch von Modellzeit steuern.

3. HSMOD- Erweiterung zur Nachbildung von Modula- Programmen

Zunächst muß hier festgestellt werden, daß nur prozeßorientierte Programme in die Betrachtung einbezogen werden. Das Hauptprogramm (Anwendungsmodul) wird als ausgezeichneter Prozeß im Modell verhandelt. Die entsprechende Erweiterung von HSMOD zeigt

schematisch Abb.2.

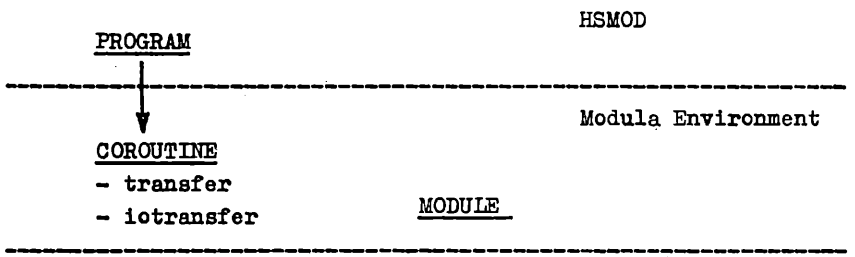


Abb. 2 Bausteine von Modula Environment

Diese Vergehensweise wird derzeitig bei der Entwicklung des lokalen Rechnernetzes an der Sektion Mathematik der HUB erprobt. Im Vortrag wird darauf näher eingegangen.

4. HSMOD- Erweiterung zur Nachbildung von Concurrent Pascal-Programmen

Neben der Nachbildung von Konstruktionen wie RECORD, MONITOR, CLASS und PROCESS müssen bei der Nachbildung von Concurrent Pascal- Programmen auch Vorkehrungen für das Laufzeitsystem - Modell getroffen werden. Abb. 3 zeigt die entsprechenden Erweiterungen.

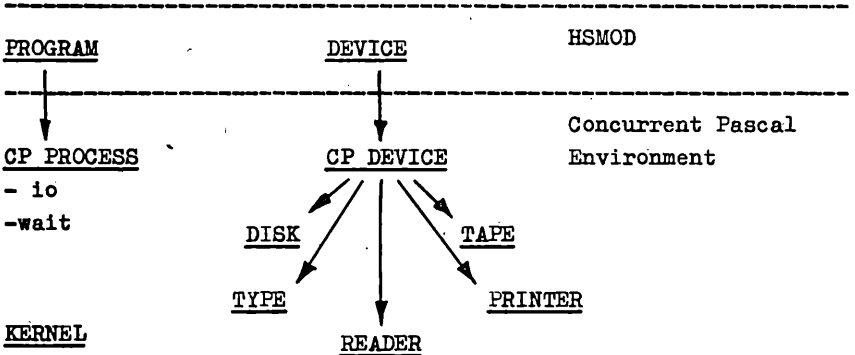


Abb. 3 Bausteine von Concurrent Pascal Environment

In /7/ findet man ein Beispiel für ein auf dieser Basis modelliertes Concurrent Pascal-Programm.

5. Literatur

/1/ Carl, D.: "Rechnergestützte Verhaltensmodellierung und -bewertung komplexer Rechnersysteme - Studie". ZfR-Informationen, ZfR- 84.03, Berlin, 1984.

/2/ Brinch-Hansen, P.: "Concurrent Pascal Report". California Institute of Technology, 1975.

/3/ Wirth, N.: "MODULA-2", ETH Zürich, Berichte des Institutes für Informatik, Nr.36, 1980.

/4/ Ahrens, K.; Fischer, J.: "First steps on modeling and simulation of Conncurrent Pascal and Modula programs". Humboldt-Universität zu Berlin, Preprint der Sektion Mathematik,Nr.77, 1984.

/5/ Dahl, O.J., Myhrhaug, B.;Nygaard, K.: "The SIMULA Common Base Language". Norwegian Computing Centre, Oslo, 1982.

/6/ Unger, B.W.; Bidulock, P.S.; Parker, J.R:: "OASIS 3.0, Reference Manual. Department of Computer Science, University of Calgary, Research Reoprt No. 81/58/10, 1981.

/7/ Fischer, J.: "Verhaltensanalyse von Concurrent Pascal Programmsystemen mittels Computersimulation". Technische Universität Dresden, Studientexte Algorithmische Sprachen ALGOL, Heft 68/84, S.22 - 28.

Dr.rer.nat. J.Fischer
Humboldt-Universität zu Berlin
Sektion Mathematik, Bereich IV
1086 Berlin, PSF 1297

Gugel, Gerhard:

"Konstruktion eines Dateninterface zur Kopplung funktionsorientierter Bausteinprogramme"

Geforderte Variabilität von DV-Lösungen (Anwendungssystemen) ist erreichbar, wenn funktionsorientierte Bausteinprogramme kombiniert werden können ohne globale Auswirkungen als Folge ihrer Kopplung. Anerkannte Strategie ist informationelle Kopplung über Daten-Schnittstellen, triviale Konstruktionsregel für das Interface ist Bilden der Vereinigungsmenge der zu transferierenden Informationen, das führt aber zu Interfaces mit hoher Redundanz.

Verbesserung durch Systematisieren und Generalisieren der Funktion der Information für ihre Verarbeitung ist möglich. Typische Funktionen abstrahierbar in Teile des Dateninterface (1: Identifikationsteil; 2: Werteteil; 3: attributiver Teil). Besondere Bedeutung hat Trennung 1: von 3: vor allem für Aufgaben der sog. "kommerziellen DV" (Gruppenmerkmale in 3: 1).

Für die konkrete Aufgabenstellung "Innovation von (Stapel-) Programmen wegen sich ändernder Nutzeranforderung und neuer technologischer Angebote (Datenbankspeicherung, Dialogverarbeitung ...)" wurde ein konkretes Dateninterface (wird vorgetragen) konstruiert und praktiziert für DV-Projekte der technologischen Vorbereitung, Planung und Kontrolle der Bauproduktion.

Erkenntnis: möglich wurden

- Koppeln von Bausteinen nur im geforderten Funktionsumfang
- Trennung (speicherkonkreter) Datenbasis von (speicherneutraler) Programmbasis (Speicherung von Verarbeitung)
- schrittweise Innovation bislang komplexer Programme
- stärkere Arbeitsteilung bei Entwicklung

Problem: noch keine sichere Aussage, ob Reduzierung des Entwicklungsaufwandes erkauft wird durch höheren Abarbeitungsaufwand wegen häufiger Exporte/Importe aus/in Standard-Dateninterface.

Erfahrungen mit der PSU auf der ES 1022

Die PSU (Projektierungs - und Programmstützende Umgebung) ist ein Softwareprodukt des Leitzentrums für Anwendungsforschung, das dem Nutzer eine UNIX - ähnliche Schnittstelle auf der Grundlage des OS/ES, sowohl in einer Stapelverarbeitungs - als auch in einer TSO - Variante bereitstellt.
Das Betriebssystem UNIX, das hier nachgebildet wird, gehört zu den verbreitetsten Betriebssystemen für Klein - und Mikrorechner, das sich durch eine moderne Architektur, komfortable Leistungen und eine effiziente Dateiverwaltung (Filesystem) auszeichnet.
Am ORZ der Humboldt-Universität ist die PSU seit Anfang 1984 im Einsatz, zunächst in einem Testbetrieb, dann zunehmend unter Einbeziehung weiterer Nutzer. Weiterhin wurde es für Praktika und andere Ausbildungsaufgaben eingesetzt.
Vorwiegend wird im Dialogbetrieb unter TSO mit der PSU gearbeitet und hat sich für die Aufgabengebiete Programmentwicklung und Verwaltung der Nutzerdateien als besondere wirkungsvoll erwiesen. Die PSU kommt mit ihrer Architektur und ihrem Kommandosatz einer Dialogarbeit besonders gut entgegen. Die Möglichkeit, mit denselben Werkzeugen und Kommandos, in derselben Umgebung die Arbeit außerhalb der Dialogzeiten im Stapel fortzuführen und zu ergänzen führt zu einer Erhöhung der Effektivität der Arbeit bei geringem Aufwand. Der Nutzer muß sein Arbeits - und Testregime nicht auf den Stapel umstellen, wie es bei paralleler Nutzung von TSO und Stapel sonst oft erforderlich war.
Im wesentlichen wurde bei der Arbeit mit der PSU wie folgt vorgegangen: Quelltext - und Objektmodulbibliotheken sowie andere nicht formatgebundene Daten wurden in das Filesystem der PSU übernommen. Dieses Filesystem bietet durch eine platzsparende Abspeicherung, den direkten Zugriff zu beliebigen Datenbytes und vor allem durch seine Reorganisationsfreiheit, die zum Beispiel die sonst bei untergliederten Dateien des OS/ES von Zeit zu Zeit notwendigen Kompreßläufe mit all ihren Gefährdungen (Sytemabsturz, Maschinenfehler, Data Checks) überflussig macht, einen sehr großen Komfort bei einfacher Handhabung.

Überhaupt muß betont werden, daß die PSU gegenüber unvorhergesehenen Unterbrechungen der Arbeit bemerkenswert resistent ist. In der Regel befindet sich das Filesystem in einem definierten Zustand, der eine Weiterarbeit an der Unterbrechungsstelle bzw. an einem unmittelbar davorliegenden Punkte ermöglicht.
Hervorzuheben ist der Editor der PSU, der durch die Vielzahl seiner Kommandos, der Schnelligkeit, mit der der editierte Text in das Filesystem zurückgeschrieben wird, gegenüber vergleichbaren Werkzeugen, zum Beispiel dem EDIT - Kommando des TSO deutliche Vorteile hat.
Die Quelltexte können in der PSU - Umgebung eingegeben, editiert und übersetzt und der Objektcode für die spätere Verbindung im Filesystem abgelegt werden. Durch das Kommando LINK ist die Verbindung der Objektmoduln zu ausführbaren Programmen möglich. Unter Berücksichtigung einiger Bedingungen ist es auch möglich, die erstellten Lademoduln in der PSU - Umgebung aufzurufen.
Fast alle Aufgaben in einer Dialogsitzung können in der PSU - Umgebung ausgeführt werden, sodaß es kaum noch notwendig ist, in die TSO - Umgebung zurückzukehren. Hinzu kommt, daß die wichtigsten TSO - Kommandos (ALLOCATE, FREE und andere) ebenfalls in der PSU aufgerufen werden können.
Allerdings ist es zur Zeit noch so, daß Lademoduln zwar in der PSU - Umgebung verbunden, aber nur in einer OS - Lademodulbibliothek abgelegt werden können. Wünschenswert wäre hier, daß auch die Lademoduln im Filesystem gespeichert werden können. Auf diese Weise wäre der Nutzer gänzlich von OS - Dateien unabhängig.
Die PSU bietet mit den Kommandos EXP und IMP (exportieren und importieren von Files) Möglichkeiten, Daten zwischen OS - Dateien und dem Filesystem auszutauschen, so daß ein Verlassen der PSU und eine Weiterarbeit im TSO - oder Stapelregime prinzipiell möglich und leicht zu bewerkstelligen ist.
Spezifische Kommandos dienen zur Sicherung einzelner Files oder des gesamten Filesystems auf Magnetband und gegebenenfalls deren Wiederherstellung.
Quelltexteditierung und Programmentwicklung, die den Schwerpunkt der Arbeit der Nutzer am ORZ der Humboldt-Universität bilden, können ausschließlich in der PSU - Umgebung durchgeführt werden.

Das Kommandokonzept der PSU, die Möglichkeit, eine Anzahl von Kommandos in Prozeduren zusammenzufassen sowie Drucklisten in die OS-Ausgabeklassen zu delegieren bieten ein umfangreiches und handliches Spektrum für diesen Aufgabenbereich.
Der PSU - Kommandosatz wird vom Hersteller und interessierten Anwendern laufend ergänzt. So sind zum Beispiel leistungsfähige Kommandos zur Textverarbeitung und - aufbereitung vorgesehen.
Hinzuweisen ist darauf, daß ein Nutzer, der mit der PSU zu arbeiten gewohnt ist, ohne Schwierigkeit sich in die Anwendung eines UNIX - Betriebssystems auf einem Klein - oder Mikrorechner einfindet. Mit zunehmender Bedeutung und Verbreitung der Mikrorechentechnik, die unter Umständen die Arbeit auf verschiedenen Rechnern und Rechnertypen erzwingen, ist dies ein nicht zu unterschätzender Vorteil. Der Nutzer ist in der Lage, mit den gleichen Mitteln und Werkzeugen, die er bereits kennt, seine Aufgaben auf einem anderen Rechner zu lösen.
Einen besonders hervorzuhebenden Gewinn stellt die Bereitstellung des C - Kompilers für die PSU durch die Technische Hochschule Karl-Marx-Stadt dar. Dieser ermöglicht nicht nur, auf einem höheren, sehr flexiblen Sprachniveau in die PSU nach Bedarf eigene Kommandos einzufügen - wobei erwähnt werden soll, daß die Schnittstellen zur PSU einfach und wohldefiniert sind - sondern setzt den Nutzer auch in die Lage, portable Programme herzustellen. Ein C - Kompiler ist in jedem UNIX - oder UNIX - ähnlichen Betriebssystem zu finden, UNIX selbst ist zum größten Teil in C geschrieben. Ein C - Text, der in der PSU ein ausführbares Programm liefert, dürfte auch auf jeder Anlage mit einem entsprechenden Betriebssystem ohne Umstände übersetzbar sein und dasselbe Programm, angepaßt an die veränderte Maschinenumgebung erzeugen. Dadurch ist Maschinenunabhängigkeit und Portabilität in einem bisher nicht vorhandenem Maße gegeben.
Die Sprache C ist leicht erlernbar und verbindet den Komfort höherer Programmiersprachen mit den Möglichkeiten einer im gewissen, die Portabilität nicht beeinträchtigenden Maße maschinennahen Programmierung. Es wird ein Code erzeugt, dessen Verhältnis zum Assemblercode in der PSU etwa 1 : 1,3 beträgt.

Die Schnittstellen zur Maschinenumgebung sind exakt definiert, es ist die Aufgabe des jeweiligen Kompilers, den für den konkreten Rechner geeigneten Code daraus zu erzeugen.
Die PSU läuft am ORZ der Humboldt-Universität auf einem ES 1022 mit 512K Hauptspeicher bzw. in einer TSO - Region von 128K.
Nach einjähriger Arbeit mit der PSU kann eingeschätzt werden, daß die Installation einfach und reibungslos verlief (ausgenommen einiger hardware-bedingten Schwierigkeiten, die in den aktuellen PSU - Versionen behoben sind). Die PSU ist inzwischen zu einem unentbehrlichen Werkzeug geworden.

ORZ der
Humboldt - Universität
108 Berlin
Unter den Linden 6

Roland Henkel

Herrlich, O.; Schubert, W.; Altmeyer, K.

KOKOZENT - ein Versuch zur Koordinierung der Entwicklung von Mikrorechnersoftware

An der Sektion Informationsverarbeitung der TU-Dresden wurde ein Konsultations- und Koordinierungszentrum für Mikrorechnersoftware im MHF-Bereich unter besonderer Berücksichtigung der Automatisierung wissenschaftlicher Experimente geschaffen.

Im Vortrag werden Ausführungen zu geplanten Arbeitsweisen und -etappen sowie zu ersten dabei erreichten Ergebnissen gemacht. In diesem Zusammenhang werden folgende Aspekte zur Koordinierung der Entwicklung und Nachnutzung von Softwaresystemen betrachtet:

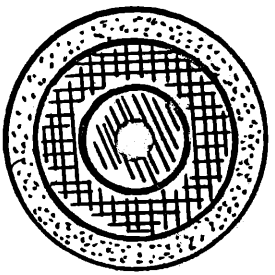
1. Welche Voraussetzungen müssen in der Planungsphase von Automatisierungsvorhaben geschaffen werden, um eine Koordinierung zum Zwecke der Rationalisierung zu erreichen?
2. Wie muß die vorhandene Software spezifiziert sein, um eine Nachnutzung für andere Automatisierungsvorhaben zu gewährleisten?
3. Welche Anforderungen an Softwarewerkzeuge zum Erstellen und Verwalten der Spezifikationen solcher Softwaresysteme stehen bei der Lösung des Koordinierungsproblems ?

Dr. Jacob, Hans-Jürgen

Projektierungsumgebung für den CCS-Rechnerverbund

1. Vorbemerkung

Der CCS-Rechnerverbund basiert auf dem Programmsystem Computer Communication Software des VEB Leitzentrum für Anwendungsforschung Berlin. In der Version CCS3 wird der Aufbau von heterogenen Rechnerverbundsystemen auf Grundlage der gegenwärtig in der DDR verfügbaren Datendienste (Handvermitteltes Datennetz, überlassene Übertragungswege) unterstützt (siehe /1/).
Die funktionale Struktur des CCS-Rechnerverbunds zeigt Abb. 1.



CCS-Datenübertragungssystem:	Synchrone Datenübertragung zwischen Rechnern im Konkurrenzbetrieb (BSC1)
CCS-Netzdienste:	Transfersysteme für Basiskommunikationsfunktionen (Netzdateikopierdienst, entfernte Bedienerkommunikation)
CCS-Projektierungsumgebung:	Gesamtheit der Mittel für die Unterstützung der Projektierung von CCS-Anwendungslösungen
CCS-Anwendungslösungen:	Universelle (z.B. entfernte Auftragsabarbeitung), spezielle (z.B. verteilte Textverarbeitung) oder bereichsspezifische (z.B. Dateitransferservice im VE KDV) verteilte Anwendung im CCS-Rechnerverbund

Abb.1: Funktionale Struktur des CCS-Rechnerverbunds

Die Projektierungsumgebung für den CCS-Rechnerverbund steht EDV-Projektanten für den Aufbau verteilter Anwendungen zur Verfügung. In folgendem sollen einige wesentliche Aspekte der Projektierungsumgebung für die Version CCS 3.1 (heterogener Verbund von ESER-EDVA und Bürocomputern) erläutert werden.
Die programmtechnische Realisierung der Projektierungsumgebung für CCS 3.1 erfolgt in Form der Datei-Dienst-Programme NFSCOPY (ESER-EDVA) /2/ bzw. NFSC (BC) /3/.

2. Kopierfunktionen der CCS-Projektierungsumgebung

Die Anwendung von Datei-Dienstprogrammen für das Umspeichern/Kopieren von Dateien ist sowohl für den EDV-Projektanten im OS/EC (z.B. IEBCOPY, IEHMOVE) als auch im SIOS 1520 (z.B. LBSV, COPD) eine gewohnte Arbeitsweise. Dem wird mit der Bereitstellung von Kopierfunktionen als Kernstück der CCS-Projektierungsumgebung Rechnung getragen. Da sich im Falle verteilter EDV-Projekte Eingabe- und Ausgabedateien auf Datenträgern lokaler und entfernter Rechner befinden, ist im Unterschied zu lokalen Datei-Dienstprogrammen die Übertragung einer Zwischendatei zwischen den Rechnern notwendig (Abb. 2).

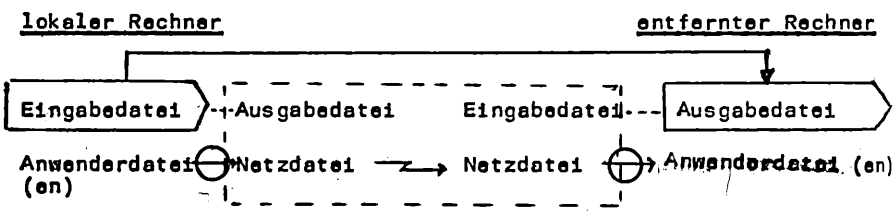


Abb. 2: Prinzipieller Ablauf des Kopiervorgangs im CCS-Rechnerverbund

Diese Zwischendatei besitzt ein netzeinheitliches Format (Netzdatei). Das CCS-Kommunikationssystem kennt nur diese Netzdateien. Entsprechende Dienste und Protokolle nehmen keinen Bezug auf die Dateien der Anwender.
Die Beschreibung der Kopierfunktionen erfolgt getrennt nach den Kommunikationsformen

ESER → ESER/ESER → BC/BC → ESER/BC → BC

Folgende Dateiorganisationsformen werden unterschieden:

PS-Dateien: Dateien, die von der OS-Datenverwaltung als sequentielle Dateien aufgefaßt werden

DA-Dateien: Direkt organisierte Dateien des OS mit und ohne Schlüssel

PO-Dateien: Untergliederte Dateien des OS

SIOS-Dateien: Sequentiell geschriebene Dateien auf Minidiskette oder Magnetbandkassette, auf die sequentiell oder direkt zugegriffen werden kann.

In der CCS-Projektierungsumgebung unterstützte Kopierfunktionen sind:

I. ESER ⟶ ESER

1. PS-Dateien:
   - einfaches Kopieren
     z.B. . Kopieren einer Banddatei
          . Kopieren einer sequentiell organisierten Plattendatei
   - Kopieren zwischen Magnetband und Magnetplatte
     z.B. . Kopieren einer lokalen, sequentiell organisierten Plattendatei in eine entfernte Banddatei
   - Kopieren mit Umblocken (nur bei RECFM=F oder FB)
     z.B. Kopieren einer Banddatei mit Verkleinerung der Blocklänge
   - Kopieren von Beständen untergliederter Dateien
     z.B. Kopieren eines Jobs aus einer lokalen Jobbibliothek in eine entfernte Banddatei
   - Kopieren von Dateien im Systemeingabestrom
     z.B. Kopieren von über Lochkartenleser eingegebenen Lochkarten in eine Banddatei
   - Kopieren von über Direktleser eingegebenen Dateien
     z.B. Kopieren von Lochkarten in eine sequentiell organisierte Plattendatei

2. DA-Dateien:
   - einfaches Kopieren
     z.B. Kopieren eine Plattendatei mit Schlüssel
   - Kopieren mit Umblocken (nur bei RECFM=F oder FB)
     z.B. Kopieren einer Plattendatei ohne Schlüssel mit Verdopplung der Blocklänge

3. PO-Dateien:
   - einfaches Kopieren
     z.B. Kopieren einer lokalen Bibliothek in eine neu zu erstellende entfernte Bibliothek
   - Kopieren aller Bestände einer PO-Datei
     z.B. Ergänzung einer entfernten Bibliothek um die Bestände einer lokalen Bibliothek
   - Kopieren von Beständen einer PO-Datei
     z.B. . Kopieren ausgewählter Bestände einer lokalen Bibliothek in eine entfernte Bibliothek
     . Kopieren von Beständen einer lokalen Bibliothek in eine entfernte Bibliothek unter Ausschluß von Beständen
   - Kopieren mit Ersetzen von Beständen
     z.B. Aktualisieren einer entfernten Bibliothek durch Änderungsstände einer lokalen Bibliothek
   - Kopieren mit Umbenennen von Beständen
     z.B. Einbringen von Beständen einer lokalen Bibliothek mit neuen Namen in eine entfernte Bibliothek

II. ESER → BC

1. PS-Dateien:
   - Kopieren einer PS-Datei in eine SIOS-Datei auf Minidiskette
     z.B. Kopieren einer lokalen Banddatei in eine entfernte SIOS-Datei auf Minidiskette
   - Kopieren einer PS-Datei in eine SIOS-Datei Auf Magnetbandkassette
     z.B. Kopieren von über Lochkartenleser eingelesenen Lochkarten in eine SIOS-Datei auf Magnetbandkassette

2. DA-Dateien:
   - Kopieren einer DA-Datei in eine SIOS-Datei auf Minidiskette
     z.B. Kopieren einer lokalen DA-Datei in eine entfernte SIOS-Datei auf Minidiskette zur Weiterverarbeitung mit direktem Zugriff
   - Kopieren einer DA-Datei in eine SIOS-Datei auf Magnetbandkassette
     z.B. Kopieren einer lokalen Plattendatei in eine entfernte SIOS-Datei auf Magnetbandkassette zur Weiterverarbeitung mit sequentiellem Zugriff

3. PO-Dateien:
   - Kopieren von Beständen untergliederter Dateien als sequentielle Dateien
     z.B. Kopieren eines Bestands einer lokalen Bibliothek in eine SIOS-Datei auf Minidiskette

Das Kopieren einer lokalen Datei im DKOI-Kode in eine entfernte Datei im KOI7-Kode ist bei allen Kopierfunktionen ESER⟶BC möglich.

III. BC⟶ESER

- Kopieren einer SIOS-Datei in eine DA-Datei
  z.B. Kopieren einer lokalen SIOS-Datei auf Magnetbandkassette in eine entfernte Plattendatei
- Kopieren einer SIOS-Datei in eine PS-Datei
  z.B. Kopieren einer lokalen SIOS-Datei auf Minidiskette in eine entfernte Banddatei

Das Kopieren einer lokalen Datei im KOI7-Kode in eine entfernte Datei im DKOI-Kode ist bei allen Kopierfunktionen BC⟶ESER möglich.

IV. BC⟶BC

- einfaches Kopieren
  z.B. Kopieren einer lokalen SIOS-Datei auf Minidiskette in eine entfernte, existierende Datei auf Minidiskette
- Kopieren zwischen Minidiskette und Magnebandkassette
  z.b. Kopieren einer lokalen SIOS-Datei auf Minidiskette in eine entfernte, existierende SIOS-Datei auf Magnetbandkassette

Die entsprechenden NFSCOPY- bzw. NFSC-Läufe können auch auf nicht am CCS-Datenübertragungssystem angeschlossenen ESER-EDVA bzw. Bürocomputern durchgeführt werden (offline Auftragserstellung). Die Übertragung der als Zwischendatei entstehenden Netzdateien erfolgt nach deren Erzeugung automatisch, jedoch hängt der Übertragungszeitpunkt von der Verfügbarkeit der entsprechenden Verbindung zwischen lokalen und entferntem Rechner ab.
Für den Abschluß des Kopiervorgangs auf Empfangsseite (Erstellen Ausgabedatei aus Netzdatei) ist wiederum die Abarbeitung der Datei-Dienstprogramme NFSCOPY bzw. NFSC notwendig. Die dafür benötigten Informationen können dem NFS-Protokoll (ESER) bzw. NFSC-Protokoll (BC) entnommen werden. Im Falle periodisch abzuarbeitender EDV-Projekte werden entsprechende NFSCOPY-Jobs bzw. NFSC-Menues bereits am Empfangs-Rechner existieren.
Bei Nutzung der Kopierfunktionen ESER ⟶ ESER besteht darüberhinaus die Möglichkeit der Übergabe des NFSCOPY-Jobs gemeinsam mit den zu kopierenden Dateien.

3. Netzdateisteuerfunktionen der CCS-Projektierungsumgebung

Diese Funktionen beziehen sich auf 3 Komplexe:

1. Festlegung von ESER-EDVA/BC, auf denen die Netzdatei bereitgestellt werden soll

- Bereitstellung auf einem bestimmten Rechner
- Bereitstellung auf mehreren Rechnern
- Bereitstellung auf allen über das CCS-Datenübertragungssystem erreichbaren Rechnern (Rundschreiben)

} nur ESER→ESER bzw. ESER→BC

2. Verringerung des zu übertragenden Datenvolumens der Netzdatei

- Standardkomprimierung
- Attributkomprimierung
- Schnittstelle für den Anschluß weiterer Komprimierungs-/Dekomprimierungs-Moduln

} nur ESER ⟶ ESER

3. Verwaltungsfunktionen

- Realisierungs- und Verfallsfristen
- Zuordnung zu einer Abrechnungsnummer
- Dateisicherung

Die gegenwärtig verfügbaren Datendienste der DP erfordern eine konsequente Minimierung des zu übertragenden Datenvolumens.
Leistungsfähige Komprimierungsverfahren, wie die dateistrukturabhängige Attributkomprimierung /4/, besitzen daher im Rahmen der CCS-Projektierungsumgebung einen besonders hohen Stellenwert.

Literatur

/1/ Rechentechnik/Datenverarbeitung
22(1985) 1, S. 5-18

/2/ CCS 3.1 Anwendungsprojektierung für den Rechnerverbund
Teil I: Anwendungsprojektierung auf ESER-EDVA
VEB LfA Berlin, HA F1, 1985

/3/ CCS 3.1 Anwendungsprojektierung für den Rechnerverbund
Teil II: Anwendungsprojektierung auf Bürocomputer
VEB LfA Berlin, HA F1, 1985

/4/ CCS 3.0 Attributkomprimierung
VEB LfA Berlin, HA F1, 1984

Dr.Ing.
Jacob, Hans-Jürgen
VEB Leitzentrum für Anwendungsforschung Berlin
Betrieb des Volkseigenen Kombinats Datenverarbeitung
1156 Berlin
Jacques-Duclos-Str. 50/52

Kiss, Gábor: GUTS- ein neues Dialogsystem

Das Dialogsystem GUTS - Gothenburg University Timesharing System - wurde entwickelt, um das TSO System zu ersetzen, da es gegenüber dem bekannten TSO-Betrieb zahlreiche Vorteile bietet. Die Anwendung spezieller Datenorganisationsmethoden auf Systemplatten und eine zweckmassig entwickelte Zugriffsmethode für die Datenübertragung zwischen der Zentraleinheit und den Terminals ermöglicht die bessere Ausnutzung der Ressourcen und die schnellere Antwortzeiten an den Terminals.

Bedingungen der Inbetriebsetzung des GUTS

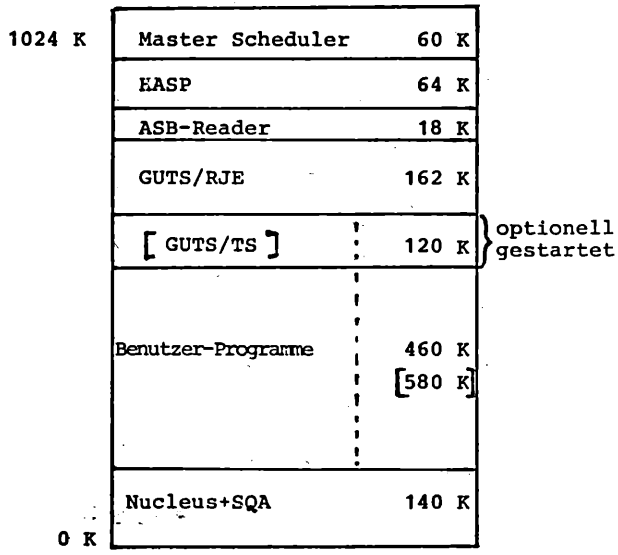
Das Dialogsystem arbeitet grundsatzlich unter der Steuerung des OS. Bei den Systemen MFT, MVT und SVS ist die Anwesenheit des HASP Untersystems unbedingt erforderlich, das das Eingabe-Ausgabe-Spooling ermöglicht und zusatzlich eine interne Reader Funktion anbietet.

Als Terminal können die IBM 3270-Gerate - lokale und entfernt - verwendet werden.

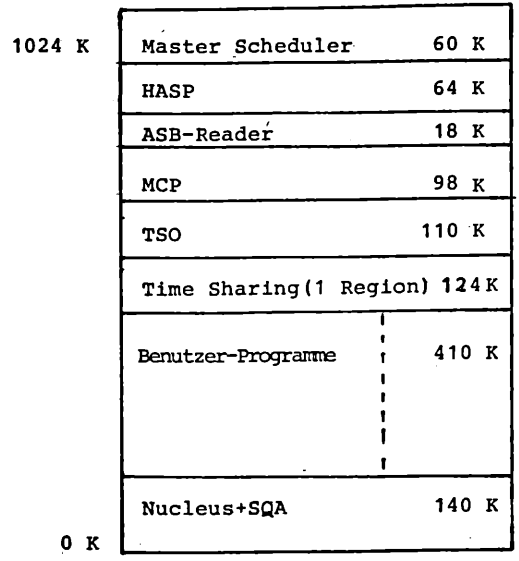
An der Budapester Universitat ist eine ESER-I Anlage /R40/ in Betrieb, die unter der Steuerung von OS/MVT+HASP+GUTS arbeitet. Die Aufteilung der Zentralspeicher ist auf der untenstehenden Abbildung dargestellt.
Zur Zeit sind 8 MERA lokal Terminals (EC 7910) an die Anlage angeschlossen.

AUFTEILUNG DES HAUPTSPEICHERS (1 Mbyte)

(R40 OS/MVT + HASP)



GUTS - Betrieb



TSO - Betrieb

Der Aufbau und die Dienstleistungen des GUTS

Das System GUTS arbeitet als selbstandige Aufgabe im Betriebsystem OS /genauso wie HASP/. Dabei müssen die folgenden Dateien on line erreichbar zu sein:

GUTS.LIBRARY - Bibliothek der Lademodulen, muss mit der Datei SYS1.LINKLIB verkettet sein

GUTS.RORI - Rollin /Rollout Datei - Benötigt GUTS/TS /Entspricht SYS1.SWAP bei TSO/

GUTS.USERDIR - enthalt die Attribute der berechtigten Benutzer /entspricht SYS1.UADS bei TSO/

GUTS.FILEDIR - praktisch das Verzeichnis der GUTS.LIBRARY

GUTS.LIBRARY - Bibliothek, die die Benutzerbestande enhalt

Ausser der obenerwahnten Dateien gibt es noch zahlreiche Hilfsdateien, die die bequeme Arbeit unterstützen /Z.B. HELP Funktion/, aber der Einfachkeit halber hier nicht behandelt werden.
Um das Konzept des Systems besser zu verstehen soll der Aufbau der GUTS.LIBRARY betrachtet werden.

Diese Bibliothek hat eine spezielle Datenorganisation. Die Benutzerbestande werden in verdichteter Form gespeichert wodurch der Plattenbereich cca 25% weniger belastet wird. Neben der Erhöhung der Datenübertragungsgeschwindigkeit hat diese spezielle Datenorganisation noch einen sehr wichtigen Vorteil, und zwar kann die Verdichtung der Bibliothek /COMPRESS INPLACE/ völlig vermieden werden.

Die Bestande der Bibliothek haben einen Namen in folgender Form: USERID.MEMNAME. Die logische Satzlange innerhalb eines Bestandes kann 1-132 Byteanzahl sein, dazu kommen noch 2 Bytes, die für eine interne Zeilennumerierung verwendet werden. Physisch werden die einzelnen Rekords /Zeilen/ in variabler Byteanzahl gespeichert, abhangig von den Verdichtungsmöglichkeiten. Auf dem Bildschirm können die Zeilen der Benutzerbestande mit oder ohne interne Zeilennumerierung ausgeschrieben werden.

Die interne Zeilennumerierung ist ein grudsatzliches Hilfsmittel im Editmodus.
Die Benutzerbestande können Quellmodulen oder sogenannten exec-procedure-Modulen sein. Als Benstandteil hat das System GUTS eine selbstandige Prozeduresprache, die auf viele Arten parametrisiert werden kann und die Bildung internen Zyklen ermöglicht. Mit Hilfe der "token", Systemvariablen und GUTS-Kommandos können auch solche ausführbare Prozeduren geschrieben werden, dei keine interaktive Region benötigen.

Da die Datei GUTS.LIBRARY ein Zentraler Informationsspecher ist, kann sie eine mehrbandige Datei sein /max. 16. Platten/.
Wie die Abbindung darstellt, befinden sich - als residenter Teil des Systems - im normalen GUTS-Betrieb die Modulen GUTS/RJE und GUTS/TS im Zentralspeicher.
Der erste Modul bietet die folgenden Dienstleistungen:

- Edit-Modus /Zeilenorientiert oder full screen editor/
- Delegierung von Jobs über den HASP-internen Reader zum OS
- Disposition über die Ergebnisse der delegierten Jobs.
  Die druckbaren Ausgabedateien können auf Bildschirm, auf zentralen Druckern oder in den GUTS-Bestand dirigiert werden.
  /Druckzeile max. 132 Charakter/
- Abfragemöglichkeiten über den Bearbeitungsstand der delegierten Jobs und über den Systemzustand von OS, HASP und GUTS.

Der zweite Modul /GUTS/TS/ ermöglicht die interaktive Arbeit, und enthalt den oder die nötigen Benutzerprogrammbereiche.
Die Anzahl und Grösse der Benutzerprogrammbereiche werden wahrend der Generierung festgelegt, aber sie können beim Start des Systems vom Bediener verandert werden. Der Inhalt dieses Bereiches - wenn gleichzeitig von mehreren Benutzern beansprucht - wird in die Dabei GUTS.RORI aus- un eingelagert.

Beim Start des GUTS wird nur der Modul GUTS/RJE in den Hauptspeicher geladen. Der zweite Modul - der vom Bediener durch ein spezielles GUTS-Kommando gestartet werden kann - wird entsprechend Benutzeransprüche gestartet.
An der Universitat ist GUTS/RJE ganztatig in Betrieb, dagegen GUTS/TS nur einige Stunden, - je nach Vorbestellung durch die Benutzern.

Im interaktiv Betrieb stehen alle Programmiersprachen zur Verfügung /FORTRAN, PL1, PL1 Optimizer, COBOL, BASIC, ASSEMBLER/, aber auch in GUTS/RJE - im Editmodus - können die entsprechenden Syntaxchecker verwendet werden.
Da die GUTS.LIBRARY eine spezielle Datenorganisation hat, können die einzelnen Bestande mit Hilfe der gewöhnlichen OS-Dienstprogramme nicht erreicht werden. Um die gespeicherten Informationen auch im Stapelbetrieb gut behandeln zu können, wird mit dem GUTS ein Dienstprogrammpaket geliefert. Diese Programme ermöglichen die Modifizierung und Übersetzung der GUTS-Bestande sowohl als batch als auch interaktiv.

Datenschutz und - sicherung

Jede Bildschirmsitzung beginnt mit der Angabe der Benutzerkennzeichnung und des Kennwortes. /USERID, PASSWORD/.
Die Eingabe der einzelnen und berechtigten Benutzerkennzeichnungen wird vom Systembetreuer durchgeführt. Der Systembetreuer wird im Zuge dieses Prozesses diejenige Möglichkeiten bestimmen, die dem Benutzer zur Verfügung stehen.
Das Kennwort kann vom Benutzer jederzeit verandert werden, aber man muss vorsichtig sein, da das Vergessen des Kennwortes alle weitere Arbeit unmöglich macht. In diesem Fall gibt es nur eine Möglichkeit, die Benutzerkennzeichnung zu löschen und erneut einzugeben.
Die Benutzerkennzeichnung bestimmt eindeutig die Arbeitsmöglichkeiten des Benutzers:

- Benutzung der interaktiven Region
- Delegierung von Jobs
- Möglichkeit zur Modifizierung der Accountinformationen und der Benutzerkennzeichnungen
- Beschrankungen im Zusammenhang mit den Namen des delegierten Jobs
- den Wert der Benutzerauthoritat /siehe unten/

Der Benutzeratuhoritatswert ist ein skalar Wert von 0 bis 15.
Dieser Wert beschrankt die Arbeit im Zuge einer Sitzung folgendermassen: /grober Überblick/

AUTH=0 Einfache GUTS-Kommandos,
Herstellung und Update der eingenen Dateien
Abhangig von der Benutzerkennzeichnung ist interaktive Arbeit oder Delegierung von Jobs möglich.
Lesen OS-Dateien

AUTH=4 Abfragemöglichkeiten über den Zustand des OS und GUTS.
Informationen über OS-Dateien /z.B. LPDS/

AUTH=6 Eigene OS-Datei kann im Zuge der interaktive Arbeit als Ausgabedatei geöffnet werden.
"Fremde" GUTS-Bestande können gelesen und beschrieben werden.

AUTH=8 Beliebige OS-Datei kann gelesen /beschrieben werden /natürlich unter der Kontrolle des Schutzsystems des OS/.
GUTS-Schutzschlüssel der fremden Benutzern kann abgefragt werden

AUTH=12 Die GUTS-Instandhaltungs- und Sicherungsfunktionen können durchgeführt werden

AUTH=15 Das Terminal kann als Bedienerkonsole verwendet werden, fast alle OS-Kommandos können ausgegeben werden.

Ausser der Benutzerkennzeichnung, dem Kennwort und dem Benutzerauthoritatswert gibt es noch eine weitere Möglichkeit - den Schutzschlüssel - die Benutzerbestande vor unbefugtem Zugriff zu schützen. Der Schutzschlüssel ist eine vorzeichenlose Zahl von 0 bis 32767, und er kann beim Erzeugen - oder Umbenennen - des Bestandes definiert werden.
Wenn der Authoritatswert eines Benutzers hoch genug ist, kann er fremde GUTS-Bestande lesen und schreiben, aber bei geschützten Bestanden ist das nur nach Angabe des Schlüssels möglich.
Das Schutzsystem des GUTS is völlig unabhangig vom Schutzsystem des OS., Der Datum- und der Kennwortschutz des OS bleibt weiterhin wirksam.

Die Benutzerbestande werden in der GUTS.LIBRARY gespeichert. Die maximale Grösse dieser Bibliothek sind 16 Platten /27 oder 100 MB Platten/, und benötigt theoretisch keine Instandhaltung. Über die Verwendungshaufigkeit der einzelnen Bestande führt das System eine sehr umfangreiche Statistik, mit Hilfe deren ein

wirksames Archivierungssystem in Betrieb gesetzt werden kann. Je Benutzerbestand werden die folgenden Daten in die Datei GUTS.FILEDIR gespeichert:

- Datum der Erstellung /oder Umbenennen/
- Datum der letzten Modifizierung
- Datum des letzten Zugriff /z. B. Lesen/
- Platzbedarf in der Bibliothek.

Die zur Verfügung gestellten Dienstprogramme ermöglichen es - entsprechend den obigen Informationen - die verfallenen und zu grossen Bestande auf Magnetband oder auf Magnetplatte zu laden. Bei Bedarf können die ausgelagerten Modulen auch schnell zurückgespeichert werden.
Der Sicherheit halber - wenn der Benutzer benötigt - kann die volle Kommunikation im Laufe der Sitzung in einer Hardcopy-Datei aufbewahrt werden. Diese Datei kann spater ausgewertet werden.

Praktische Erfahrungen

An der Budapester Universitat steht eine R40 Anlage, in Betrieb, die mit acht lokalen Bildschirmterminals /EC 7910/ verbunden ist. Das betriebene Operationssystem ist OS/MVT mit HASP. Seit einem Jahr ist das GUTS eingesetzt, das das TSO System ersetzt hat. Laut den bisherigen Erfahrungen kann eindeutig ausgesagt werden, dass die Dienstleistungen des GUTS in jeder Hinsicht besser sind:
- weiniger Hauptspeicheraufwand
- Kürzere Terminalantwortzeiten /z. B. LOGON, LOGOFF, SAVE innerhalb 2 Sekunden/
- ein zuverlassigeres Datenschutzsystem. Die OS-Dateien sind nur für berechtigten Benutzer erreichbar
- der Systemkatalog des OS wird nicht belastet
- das eigene Sicherungssystem ermöglicht die bequeme und wirksame Instandhaltung der on line GUTS Dateien. Die verfallenen Benutzerbestande können systematisch und dokumentiert auf Magnetband ausgelagert werden.
- nach einen eventuellen Systemabsturz werden die aktiven Edit-Arbeitsbereiche aufbewahrt, und im Zuge einer spateren Sitzung

kann die vorherige Dateneingabe reibungslos fortgesetzt werden. /Die Aufbewahrung geschieht max. 7 Tage lang/

- die interaktiv gestarteten Programme - abhangig von der Berechtigung des Benutzers - können alle OS-Dateien erreichen, entweder in Input- oder Output-Modus.
  Die OS-Dateien können auch DSORG=DA oder DSORG=IS sein, im Gegensatz zum TSO.

Ausser den obenerwahnten Möglichkeiten gibt es noch weitere Interfaces zu anderen Telekommunikationssystemen, z. B. CICS und SHADOW.

Literatur

1/ GUTS Installation Manual Ref: AS-2003-3 1982
2/ GUTS Reference Manual Ref: AS-2001-3 1982
3/ GUTS System Manual Ref: AS-2005-2 1981.

Verfasser:

Dr. Kiss, Gábor
ELTE Számitóközpont
1117 Budapest
Bogdánffy u. 10/b.

Ingo Schreye, Reinhold Schönefeld

Programmierwerkzeuge unter dem Betriebssystem MUTOS 1630

MUTOS 1630 (multi user timesharing operating system) ist eine Entwicklung von INEUM Leningrad, VEB Kombinat Robotron und der Technischen Hochschule Ilmenau für den Kleinrechner K1630. Dieses Betriebssystem ist kompatibel zu dem Betriebssystem UNIX - Version 7 /1/.

MUTOS ist ein dialogfähiges Betriebsystem, das durch folgende Eigenschaften gekennzeichnet ist:

- ein hierarchisches Filesystem;
- die Kompatibilität zwischen Files, Geräten und der Interprozess-E/A;
- die Möglichkeit, asynchrone Prozesse zu schaffen;
- eine universelle Systemkommandosprache;
- ein hoher Grad an Portabilität;
- eine einfache Handhabung einer grossen Anzahl von Systemrufen;

Unter MUTOS sind eine Reihe Programmierwerkzeuge implementiert, die dem Nutzer eine effektivere Softwareentwicklung gestatten. Die wichtigste Programmiersprache des MUTOS 1630 ist C /2/. Fast alle Kommandos und etwa 90% des Betriebssystemkerns sind in C geschrieben. C ist eine höhere Programmiersprache, die durch ihre ökonomische Schreibweise der Ausdrücke, die effiziente Programmablaufsteuerung, die Fähigkeit, Datenstrukturen zu definieren, sowie einen grossen Satz von Operatoren und Datentypen charakterisiert ist.

Aus der Menge der vom Nutzer zu verwendenten Programmierwerkzeuge sollen hier die Sprachentwicklungswerkzeuge

näher betrachtet werden. Dies sind universelle Hilfsmittel für alle Programme mit strukturierbarer Eingabe.

Wir wollen ein Werkzeug verstehen als ein Programm, das die Maschine nutzt und ein allgemeines Problem löst. Damit viele Nutzer dieses Werkzeug verwenden, soll es leicht zu handhaben sein.

Hieraus ergeben sich folgende Forderungen an Softwarewerkzeuge:

- sie müssen gut entworfen sein, damit der Leser die Möglichkeit hat, Programme zu erstellen, die leicht zu pflegen und zu ändern sind;
- sie müssen nutzerfreundlich sein, um einen gewissen Anwenderkomfort zu gewährleisten;
- sie müssen zuverlässig sein, um die Richtigkeit der erzielten Ergebnisse garantieren zu können;
- sie müssen effizient sein, damit man sich die Programmausführung auch leisten kann;

Vorteile und Eigenschaften von Softwarewerkzeugen:

- schnellere Erzeugung von Softwareprodukten;
- generierte Programme arbeiten korrekt;
- Anwenderwünschen kann schneller Rechnung getragen werden, d.h. eine schnelle Änderung des Problementwurfs ist möglich;
- Unterstützung der Portabilität, im Hinblick auf die Lösung allgemeiner Probleme (auf verschiedenen Rechnern);
- Zerlegung des Problems in leichter zu übersehauende Teilaufgaben;
- Nutzung von ausgefeilten Algorithmen und Spezialkenntnissen, die die Voraussetzung guter Programmierwerkzeuge sind;

In diesem Vortrag soll, wie schon erwähnt, auf Sprachentwicklungswerkzeuge unter MUTOS 1630 näher eingegangen werden. Speziell werden die Programme zur Erzeugung eines syntaktischen Analysators /Yacc/ und zur Erzeugung eines lexikalen Analysators /Lex/ sowie deren Zusammenarbeit vorgestellt /3/.

Yacc - Yet another Compiler Compiler

Yacc ist ein Programmgenerator. Dieser generiert syntaktische Analysatoren (Parser) aus einer Eingabespezifikationssprache, die die gewünschte Syntax beschreibt. Yacc behandelt eine grosse Menge kontextfreier Grammatiken. In Abb. 1 ist ein einfaches Beispiel einer Yacc-Eingabegrammatik dargestellt.

```
datum   :  tag "." monat jahr;
monat   :  "Jan" | "Feb" | "März" | "Apr" | "Mai" | "Jun"|
           "Jul" | "Aug" | "Sep" | "Okt" | "Nov" | "Dez" ;
tag     :  zahl ;
jahr    :  zahl |  ;
zahl    :  ZIFFER
        |  zahl  ZIFFER
        ;
```

Abb. 1 Eingabegrammatik für Yacc

Das **datum** wird definiert als **tag** , gefolgt von einem Punkt, **monat** und **jahr**. Semikolon und Doppelpunkt sind syntaktische Zeichen zur Definition der linken Seite und der Beendigung einer Regel. Der senkrechte Strich kennzeichnet Alternativen einer Regel, die Hochkommas schliessen Literale ein. **monat** ist daher entweder "Jan" oder "Feb" u.s.w. Ein **jahr** ist entweder eine **zahl** oder es kann weggelassen werden.

In unserem Fall wäre also ein Datum

3. März 1985 oder 3. Feb .

Eine **zahl** ist entweder eine ZIFFER, oder eine **zahl**, gefolgt von einer ZIFFER. Mit jeder Grammatikregel kann eine Aktion verbunden werden, d.h. ein Programmfragment in einer bestimmten Programmiersprache. Diese Programmstücken sind in unserem Fall in C geschrieben. Jedes Element einer Regel besitzt einen Wert oder eine Bedeutung. Die Aktionen sollen den Wert oder die Bedeutung der linken Seite der aktuellen Regel berechnen. Aus diesem Grund wurde ein Mechanismus geschaffen, der diesen Programmfragmenten die Werte der Elemente der Regeln übergibt. Ausserdem kann man mit diesem Mechanismus auch den Wert der aktuellen Regel an andere Regeln übergeben. Betrachten wir die Regel für **zahl** aus unserem Beispiel. Jede ZIFFER besitzt einen Wert. Man kann nun eine Aktion einbauen, die den dezimalen Wert der **zahl** berechnet. Die Werte der Komponenten der rechten Seite der Regel werden durch die Pseudovariablen ¤1, ¤2 ... beschrieben (siehe Abb. 2). Der Wert der aktuellen Regel wird mit ¤¤ übergeben.

```
zahl    :   ZIFFER
                { ¤¤ = ¤1; }
        |   zahl  ZIFFER
                { ¤¤ = ¤1 * 10 + ¤2 ;}
        ;
```

Abb. 2 Wertübergabe mittels Aktionen

Nach dem Einfügen dieser Aktionen können alle Regeln, die die Regel **zahl** verwenden, auf den Wert der **zahl** zugreifen.

Um die Eingabezeichenfolge zu lesen und sie in lexikale Einheiten (Token) umzuwandeln, ist ein lexikaler Analysator nötig. Der lexikale Analysator wandelt z.B. eine Folge von alphabethischen Zeichen in Namen um, erkennt Zeichenklassen (z.B. ZIFFER'n) und behandelt Leerzeichen, Kommentare, Newlines und ähnliche Dinge. Solche Programme können u.a. auch mit dem Programmgenerator Lex erstellt werden. Durch das Analyseprogramm können dem aufrufenden Parser die mit den gelesenen Token verbundenen Werte übergeben werden. Wird die oben angegebene Grammatik mit Yacc übersetzt, so liest der Parser, in Verbindung mit einem lexikalen Analysator, nur Daten. Er überprüft, ob die Zeichenfolgen der Eingabe dieser Grammatik entsprechen oder ob die Eingabe fehlerhaft ist.

Dem Nutzer werden von Yacc einige Vorteile angeboten. Ihm wird die Möglichkeit gegeben, die Arbeit des Parsers zu kontrollieren, wenn ein Fehler in der Eingabe entdeckt wurde. Die Fehlerstabilisierung kann mit zusätzlichen Regeln ermöglicht werden. In diesen Regeln wird ein Token "error" verwendet, um den Punkt der Fehlerstabilisierung zu kennzeichnen. Existiert z.B. eine Regel für ein Statement, in der jedes Statement mit einem Semikolon abgeschlossen werden muss, so kann man die Überprüfung der Eingabe nach Fehlerauftritt bei diesem Semikolon fortsetzen. Die Regel dazu hätte folgendes Aussehen:

```
statement  :   error  ';'
           |   ...
```

Ein weiterer Punkt, der grosse Beachtung verdient, ist die Semantikstabilisierung, d.h., wie kann man z.B. Symboltabelleneinträge oder Ausdrucksbäume "reparieren", wenn Fehler nach diesen Einträgen aufgetreten sind. Yacc besitzt eine sehr nützliche Eigenschaft, die es dem Nutzer erlaubt, arithmetische Ausdrücke zu spezifizieren. In den meisten Sprachen gibt es eine Menge von Operatoren (z.B.

+,-,/,*,...); diesen kann durch Yacc ein Vorrang zugeordnet werden. Der Ausdruck

a + b * c

bedeutet eigentlich

a + (b * c)

da die Multiplikation höheren Vorrang als die Addition hat. Man kann einfache Regeln in der Yacc-Grammatik angeben, in denen die Vorranginformation für die einzelnen Operatoren enthalten ist.

```
%left '+' '-'
%left '*' '/'
```

bedeutet, dass Addition und Subtraktion geringeren Vorrang als Multiplikation und Division besitzen. Alle Operatoren sind linksassoziativ. Diese Eigenschaft bedingt nicht nur einfacher zu schreibende Grammatiken, sondern es werden auch schnellere und kleinere Parser generiert.

Der generierte Parser wird von Yacc in einem Unterprogramm **yyparse** untergebracht. Dieses Programm stellt einen deterministischen Kellerautomaten dar /4/, der durch die Eingabe, einen Zustandskeller und eine Zerlegungstabelle gekennzeichnet ist. Angenommen, der lexikale Analysator liefert für die auf Abb. 1 angegebene Grammatik die folgende Tokenfolge:

ZIFFER ZIFFER '.' Feb

Der Parser erhält also das erste Token ZIFFER, das er im Keller abspeichert. Aus der Zerlegungstabelle wird erkannt, das ZIFFER die rechte Seite einer Regel ist. Diese Regel wird "reduziert", d.h. ZIFFER wird ausgekellert und an dessen Stelle wird die linke Seite der Regel (**zahl**) eingekellert. Das nächste Token (ZIFFER) wird gelesen und abgespeichert. Im Keller befindet sich jetzt die Symbolfolge

zahl ZIFFER

Aus der Grammatik ist zu ersehen, dass dies die rechte Seite

der letzten Regel ist. Diese Regel wird, wie oben dargestellt, reduziert, und im Keller befindet sich nun noch das Symbol **zahl**. Tritt in der Eingabe kein weiteres Token ZIFFER auf, so wird **zahl** als rechte Seite der Regel für **tag** aufgefasst und reduziert. Vom lexikalen Analysator wird nun das Zeichen '.' (Punkt) übergeben. Dieses Symbol wird in den Keller gebracht und nach dem nächsten Token gefragt. Der Parser erhält "Feb", und da "Feb" der Regel **monat** entspricht, befindet sich nach dem Reduzieren dieser Regel die folgende Symbolfolge im Keller:

tag '.' monat

Diese Folge von Symbolen stellt die rechte Seite der Regel **datum** dar. Nach Reduzierung dieser Regel ist die Eingabe vollständig abgearbeitet. Die Eingabe wurde als gültiger Satz unserer Beispielgrammatik erkannt. Parseraktionen werden beim Erkennen einer Regel ausgeführt, d.h., bevor die Symbole ausgekellert werden, wird das mit der Regel verbundene Programmfragment abgearbeitet.

Zusammenfassend ist zu sagen, dass Yacc ein Werkzeug für die Generierung von Parsern liefert, die eine umfangreiche Klasse BNF-ähnlicher Grammatiken akzeptieren. Yacc besitzt gute Eigenschaften für die Fehlerstabilisierung, die Spezifikation von Operatorvorrängen und die Angabe von Aktionen für erkannte Regeln. Es stellt einen Programmgenerator dar, der für seine Arbeit einen lexikalen Analysator benötigt. Im folgenden Abschnitt wird ein Werkzeug zur Generierung solcher lexikalen Analysatoren vorgestellt.

Lex - Ein Programm zur Generierung lexikaler Analysatoren

Lex wird ähnlich verwendet wie Yacc. Die Eingabespezifikation für Lex ist wie bei Yacc eine Folge von Regeln und dazugehörigen Aktionen. Auch hier wird eine Aktion abgearbeitet, wenn die entsprechende Regel erkannt wurde. Der

Hauptunterschied besteht in der Art der Eingabespezifikation. Yacc akzeptiert BNF-ähnliche Regeln, die aus Token aufgebaut sind. Diese Token setzen sich wieder aus verschiedenen Zeichen zusammen, die die BNF-Beschreibung nicht beinhaltet. Lex liest die Zeichen von der Eingabezeichenfolge direkt. Die Eingabespezifikationssprache für Lex beruht auf der Theorie der regulären Ausdrücke /4/. Diese Klasse von Regeln bedingt, dass Lex lexikale Analysatoren generiert, die deterministische Kellerautomaten darstellen (also genauso wie bei Yacc). Gleichzeitig bedeutet das, dass auch für umfangreiche Eingabebeschreibungen der erzeugte Generator sehr schnell arbeitet.

Die Aktionen, die vom Nutzer geschrieben werden, werden in der Reihenfolge, in der die entsprechenden regulären Ausdrücke in der Eingabezeichenfolge erkannt werden, abgearbeitet. Lex generiert lexikale Analysatoren, die auch mehrdeutige Spezifikationen akzeptieren. Sie wählen die Regel aus, die die längste mögliche Zeichenfolge des Eingabestroms erkennt. Um z.B. bestimmte Wörter (angenommen "Bleistift") aus dem Eingabetext zu löschen, ist folgende Regel nötig:

```
%%
Bleistift  ;
```

Das Zeichen "%%" kennzeichnet den Beginn des Regelteils von Lex. Der reguläre Ausdruck entspricht hier genau dem Wort "Bleistift". Es wird keine Aktion angegeben, d.h. das Wort wird beim Erkennen ignoriert. Alle Zeichen, die nicht in irgendeiner Regel bearbeitet werden, werden direkt in die Ausgabe kopiert. Angenommen, wir wollen im obigen Beispiel noch Kugelschreiber in Füller umwandeln, so kommt noch die folgende Regel hinzu:

```
%%
Bleistift   ;
Kugelschreiber  printf("Füller");
```

Der Kellerautomat, der für diese Quelle generiert wird, sucht in der Eingabe nach beiden Zeichenfolgen (sprich Wörtern) und arbeitet die angegebenen Aktionen ab, wenn er das entsprechende Wort erkannt hat. Im allgemeinen sind die Aktionen, wiederum wie in Yacc, C-Programmfragmente. Die Zeichenfolge, die durch die Regel erkannt wurde, kann in einem Feld abgelegt werden. Dieses Feld von Zeichen heisst **yytext.** In den Aktionen kann man dieses Feld verwenden, um auf die gelesenen Zeichen zugreifen zu können.

Die Struktur der Ausgabe von Lex ist ähnlich der von Yacc. Es wird ein Unterprogramm **yylex** generiert, das den deterministischen Kellerautomaten darstellt. Yacc ruft standardmässig immer diese Funktion **yylex** auf. Jede Regel, die einen Token erkennt, muss einen Wert übergeben, der die Art des gelesenen Tokens kennzeichnet:

```
return( Tokennummer );
```

Ein Wert, der mit dem Token verbunden ist, kann der Variablen **yylval** zugewiesen werden. Vom aufrufenden Programm kann man auf diese Variable direkt zugreifen.

Ein regulärer Ausdruck kennzeichnet eine Menge von Zeichenketten, die erkannt werden sollen. Sie enthalten Textzeichen und Operatorzeichen. Die Ziffern und die Buchstaben des Alphabets sind immer Textzeichen. D.h.

```
register
```

erkennt die Zeichenkette "register" (egal, wo sie auftritt) und

```
15DD7q
```

erkennt die Kette "15DD7q". Die standardmässigen Sonderzeichen in C, wie \n (Newline), \t (Tabulator) oder \b (Backslash), können hier auch verwendet werden. Unter Lex gibt es die folgenden Operatoren mit ihren entsprechenden Verwendungsmöglichkeiten:

1. * : bedeutet die null- oder mehrmalige Wiederholung eines regulären Ausdrucks;
   + : bedeutet die ein- oder mehrmalige Wiederholung eines regulären Ausdrucks;
2. ? : stellt die alternative Auswahl von Ausdrücken dar;
3. | : Auswahl von 2 oder mehreren Mustern (Zeichenfolgen);
4. ^ : erkennt den Beginn einer Textzeile und
   ¤ : erkennt das Ende einer Zeile;
5. .(Punkt) : stellvertretend für alle Zeichen, ausser für Newline;
6. (und) : diese Klammern gruppieren Ausdrücke;
7. und " : dienen dazu, Sonderzeichen zu kennzeichnen;
8. [und] : diese Klammern definieren Zeichenklassen;
9. { und } : Zugriff auf definierte Zeichenfolgen;
10. / : Kennzeichnung von Kontextbedingungen;

Einige einfache Beispiele:

[0-9] erkennt die einzelnen Ziffern von 0 bis 9;
[0-9]+ erkennt Ziffernfolgen einer oder mehrerer Ziffern;
-?[0-9]+ erkennt Ziffernfolgen mit wahlweise vorangestelltem Minuszeichen;
[A-Za-z][A-Za-z0-9]* erkennt alphanumerische Zeichenfolgen, die mit einem Buchstaben beginnen (wird oft für das Erkennen von Bezeichnern verwendet);

Die Regeln von Lex können geringe Kontextinformationen verarbeiten. Die beiden einfachsten Operatoren hierfür sind ^ und ¤. Ist das erste Zeichen eines Ausdrucks "^", so wird der Ausdruck nur zu Beginn einer Zeile erkannt, d.h. nach einem Newline, oder am Anfang der Eingabe. Ist das letzte Zeichen ein "¤", so wird der Ausdruck nur am Ende einer Zeile erkannt. Der Kontextoperator "/" bedeutet, dass der Ausdruck

ab/cd

die Zeichenfolge "ab" nur dann erkennt, wenn ihr "cd" folgt.

Obwohl Lex für die Kompilerkonstruktion entwickelt wurde, kann es auch in anderen Fällen Verwendung finden. So z.B. als Übersetzungsprogramm oder als Teil eines portablen Optimierers. In diesem wurden die regulären Ausdrücke dazu verwendet, den Optimierer an verschiedene Assembler anzupassen /3/.

Zusammenarbeit zwischen Lex und Yacc

Da Yacc Parser generiert und Lex als Generator für lexikale Analysatoren verwendet werden kann, ist es sinnvoll, diese beiden Werkzeuge zusammenarbeiten zu lassen. Sie können als "Frontend" eines Kompilers verwendet werden. Hierzu sind allerdings zwei Dinge notwendig. Erstens eine Beschreibung der Zeichenfolgen, die die Token darstellen sollen. Diese Beschreibung wird mit Hilfe der regulären Ausdrücke gemacht. Zweitens die Beschreibung der Syntax der gewünschten Sprache in Form von Grammatikregeln. Yacc und Lex generieren aus diesen Beschreibungen die entsprechenden Programme für die lexikale und syntaktische Analyse. Diese Programme können zusammen kompiliert werden und ergeben ein abarbeitbares Programm. Dieses Programm liest die Eingabe, teilt sie in entsprechende Token auf und übergibt diese Token dem Parser. Der Parser organisiert dann die Eingabe mittels der Token in eine bestimmte Datenstruktur der gewünschten Sprache.

Das Yaccprogramm ist immer mit **yyparse** benannt und das Lexprogramm mit **yylex**. Von **yyparse** wird **yylex** aufgerufen, so dass sich die Zusammenarbeit der beiden Programme wie in Abbildung 3 darstellt.

```
                 lexikale        Grammatik-
                 Regeln          regeln
                    |               |
                    |               |
                |-------|       |-------|
                |  Lex  |       |  Yacc |
                |       |       |       |
                ---------       ---------

                    |               |
                    |               |
                |-------|       |----------|
Eingabe ------>| yylex | ----->|  yyparse | ------> Ausgabe
                |       |       |          |
                ---------       ------------
```

Abb. 3 Zusammenarbeit zwischen Yacc und Lex

Schlussbemerkungen

Schafft man in einem Betriebssytem eine Menge von Programmierwerkzeugen, so bietet das eine grosse Flexibilität und einige wesentliche Vorteile. Diese Werkzeuge unterstützen die Modularität und finden oft auch ausserhalb der für sie definierten Umgebung Verwendung. Während für die Erstellung von Werkzeugen eine gute Theorie von grossem Nutzen ist, muss man allerdings auch einige Hintertüren für solche Probleme offenhalten, die man mit diesen Theorien nicht bearbeiten kann. Es ist wichtig, sich mit den theoretischen Grundlagen zu befassen. Verwendet man z.B. ein theoretisches Modell, das sich als ungenügend erweist, als Grundlage eines Softwarewerkzeugs zu dienen, so ist es sinnvoll, diese Theorie weiterzuentwickeln, um neue und bessere Werkzeuge zu produzieren. Zum Schluss sei hier noch angefügt, dass die Erstellung von Softwarewerkzeugen mit der Verbesserung der allgemeinen Programmierumgebung unmittelbar

verbunden ist. Dieser Artikel wurde z.B. mit dem Progammierwerkzeug "nroff" erstellt. "nroff" ist ein Textverarbeitungsprogramm, das unter MUTOS 1630 implementiert ist.

Literaturverzeichnis

/1/ Ritchie,D.M.;Thompson,K.
"The UNIX Time - Sharing System"
The Bell System Techn. Jour.,Juli/Aug. 1978,
Vol. 57, No.6, part 2

/2/Kernighan,B.W.;Ritchie,D.M.
"The C Programming Language"
Prentice-Hall, Englewood Cliffs, N.J., 1978

/3/Johnson,S.C.
"Language Development Tools on the
Unix System"
IEEE Computer, Aug. 1980, pp 16-21

/4/Aho,A.V.;Ullman,J.D.
"Principles of Compiler Design"
Addison-Wesley, Reading, Mass., 1977

Prof.Dr.sc.nat. R. Schönefeld,
Dipl.-Ing. I. Schrewe
Technische Hochschule Ilmenau
DDR - 6300 Ilmenau
PSF 327

Schuster, Harald

Programmentwicklung und Datenverwaltung

1. Einleitung

Mit der steigenden Komplexität von Software werden bessere Möglichkeiten zu deren rationeller Produktion erforderlich. Es ist dabei das Ziel, diesen Entwicklungsprozeß einheitlich über alle seine Phasen, von der Spezifikation bis zur Fertigstellung der Software, einschließlich der dazugehörigen Dokumentation zu führen. In jeder dieser Phasen entstehen neue Daten, alte Daten verfallen oder müssen an die nächste Phase weitergereicht werden. Deshalb kommt einer leistungsfähigen Datenverwaltung (DV) große Bedeutung zu.

Die von den meisten Betriebssystemen angebotenen Möglichkeiten zur Datenverwaltung sind jedoch i.allg. an den physischen Speicherformen orientiert. Der Nutzer muß sich diesem Regime und seinen Datenobjekten wie Bibiiotheken, Bestände usw. unterordnen. Das entspricht aber nicht der Arbeits- und Denkweise bei der Softwareentwicklung. Dort geht es um die Entwicklung von Programmen, die z. B. aus Moduln und Routinen bestehen können sowie um deren Dokumentation. Es zeigt sich, daß es einer speziellen Datenverwaltung bedarf, um den Softwareentwicklungsprozeß aktiv unterstützen zu können /Ko82/.

Die folgenden Ausführungen befassen sich mit einigen speziellen Problemen der Datenspeicherung innerhalb von Programmentwicklungssystemen (PES). Dabei wurde besonderer Wert auf eine hohe Sicherheit für die gespeicherten Daten gegenüber Verlusten durch Betriebssystem- u. a. Fehler gelegt.

Eine Implementation einer solchen DV in der Programmiersprache PASCAL befindet sich gegenwärtig in der Entwicklung.

2. Merkmale einer Datenverwaltung für Programmentwicklungssysteme

Es gibt einige Merkmale, denen eine solche DV genügen muß:

- Handhabbarkeit:

  Es muß eine Nutzerschnittstelle existieren, die den Zugang zu den Daten auf einem hohen Niveau realisiert, die aber gleichzeitig einfach aufgebaut und leicht erlernbar ist. Das Filesystem XFS /Ro82/ bietet z.B. auf der Basis von UNIX /RT74/ eine Reihe zusätzlicher Dienste wie Sicherungsalgorithmen und Zugriffsstatistiken an, wobei weitestgehend das bekannte UNIX-Filesysteminterface beibehalten wurde. Dadurch kann der Nutzer mit bekannten Namen und Begriffen operieren, was die Akzeptanzschwelle herabsetzt.
  Für die Handhabbarkeit einer DV ist es ebenfalls wichtig, daß notwendige Wartungsarbeiten auf ein Minimum reduziert werden, so daß die Entwicklungsarbeit davon nicht beeinträchtigt wird. Das erfordert i. all. reorganisationsfreie Speicherstrukturen /HHMT84/.

- Portabilität:

  Moderne Software sollte so beschaffen sein, daß ihr Einsatz auf verschiedenen Rechnerfamilien (ESER, SKR, BC, ...) möglich ist. Die Implementation muß daher in einer Programmiersprache erfolgen, die zwei Eigenschaften besitzt:

  - Verfügbarkeit für die infrage kommenden Rechner (als Grundbedingung),
  - Unterstützung eines Modulkonzepts.

  Die zweite Eigenschaft wird gefordert, um eine Schichtenarchitektur der DV-Software zu ermöglichen. Dies ist notwendig, um eine klare Trennung der verschiedenen funktionellen Ebenen zu realisieren. Das gilt vorallem für die Schnittstellen zwischen den maschinenabhängigen und maschinenunabhängigen Teilen der Software, da dort bei einer Portierung der DV-Software oft umfangreiche Anpassungsarbeiten erforderlich sind.

- Offenheit:

Unter diesem Begriff sollen zwei Aspekte betrachtet werden: die wahlweise Nutzung von Systemkomponenten und der Datentransfer. Aufgrund der oben diskutierten Forderung nach Portabilität macht sich beim Übergang von einem Rechner zu einem anderen u. U. eine Auswahl von Teilen der DV-Software erforderlich. Das kann z. B. durch unterschiedliche Hauptspeichergrößen oder Betriebssystemkomponenten nötig sein. Eine Schichtenarchitektur unterstützt eine solche optionale Nutzung.
Offenheit bedeutet aber auch, daß einfache Wege für den Transport von Daten sowohl vom Betriebssystem in die DV als auch umgekehrt nötig sind. Das ist u. a. erforderlich, um bereits existierende Software (z. B. Compiler), die nicht mit den Zugriffsroutinen der DV arbeiten können, in das Programmentwicklungssystem mit einzubeziehen.

- Sicherheit:

Eine effektive Programmentwicklung ist nur dann möglich, wenn die gespeicherten Daten gegen Verluste und Verfälschungen durch Hard- oder Softwarefehler geschützt sind. Dabei ist es für den Nutzer wichtig, daß

- Datensicherungsalgorithmen die normale Arbeit wenig beeinträchtigen,
- Datenwiederherstellung <u>sofort</u> nach einem Fehler möglich ist und
- der Zustand der Daten <u>nach</u> einer Wiederherstellungsmaßnahme möglichst dem <u>vorherigen</u> Stand entspricht.

Die herkömmlichen Verfahren zur Datensicherung auf der Basis von Sicherungskopien und Abzügen entsprechen insbesondere nicht der letzten Forderung.

3. Interne Filestrukturen

Geht man von der Forderung nach Reorganisationsfreiheit der Daten aus, so stellt sich die DV aus der Sicht des Betriebssystems als ein Speicherraum dar, dessen Blöcke B_k mit gleicher und fester Länge l die physische Zugriffseinheit bilden. Aus der Sicht der Nutzer bzw. Nutzerprogramme besteht die DV aus einer Menge von voneinander unabhängigen Files F_i, deren Segmente S_j die logische Zugriffseinheit bilden. Vereinfachend sei angenommen, daß auch die Segmente gleiche und feste Länge besitzen. Es wird ein Abbildungsmechanismus $B_k \leftrightarrow S_j$ als Bindeglied zwischen physischen Blöcken und logischen Segmenten erforderlich. Damit kommt man zu der in Bild 1 dargestellten Speicherstruktur.

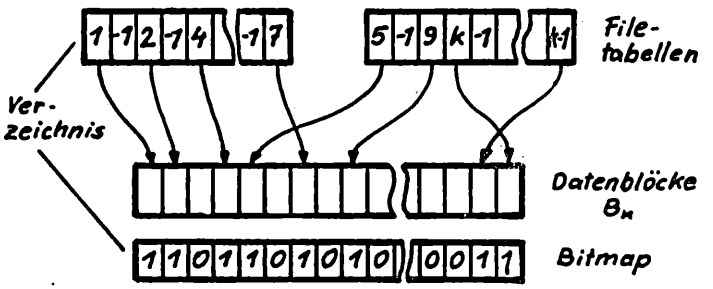


Bild 1: Reorganisationsfreie Speicherstruktur

Eine solche Grundstruktur, die typisch ist für viele DV-Implementationen, ist jedoch sehr anfällig für Systemfehler. Wenn es zu fehlerhaften Betriebssystemzuständen kommt, nachdem Daten verändert wurden, aber dieses noch nicht im Verzeichnis vermerkt ist, so geht u. U. die Integrität der gesamten DV verloren. Es entstehen Fehler in der Bitmap oder in den Filetabellen; diese führen in der Folge zu fehlerhaften Adreßauflösungen, so daß Datenblöcke entweder nicht mehr erreichbar sind oder unzulässigerweise als frei gemeldet werden.

4. Datensicherheit

4.1. Fehlerarten

Das im folgenden vorgestellte Datensicherungsverfahren ist für die Behandlung folgender Fehlerarten geeignet:

- Systemfehler (Crash):
  - plötzliche und unvorhersehbare Beendigung aller im Rechner ablaufenden Prozesse,
  - Verlust aller im Hauptspeicher befindlichen Daten einschl. Puffer,
  - undefinierte Zustände der Blöcke, auf denen gerade Schreibvorgänge aktiv waren,
  - kann entstehen durch Prozessorausfall, Stromversorgungsausfall usw.
- Plattenfehler:
  - Datenverfälschung oder -verlust eines oder mehrerer Datenblöcke,
  - kann entstehen durch mechanische Beschädigungen der Plattenoberfläche, Schmutzablagerungen, altersbedingte Veränderung der Magnetisierung usw.
- Nutzerfehler:
  - vom Nutzer gewünschter oder von der DV geforderter Abbruch einer Speicherafugabe,
  - kann erforderlich werden durch drohende Integritätsverletzung, fehlerhafte Speicheranforderung usw.

4.2. Transaktionen

Bei der rechnergestützten Softwareentwicklung handelt es sich um einen sehr datenintensiven Prozeß. Dabei wird durch die Anwendung einer datenverändernden Operation O von einem integren Datenzustand I_i zu einem anderen integren Zustand I_{i+1} übergegangen. Häufig besteht die Operation O aus mehreren Teiloperationen $O_1,\ldots,O_k$. Die dabei entstehenden Zwischenzustände sind

nicht notwendig integer. Wenn mit Z_j $(j=1,..,k-1)$ die Zwischenzustände bezeichnet werden, läßt sich die Auflösung der Operation O in eine Folge von Operationen wie folgt angeben:

$$Z_1=O_1(I_i);\ Z_2=O_2(Z_1),\ \ldots\ ,\ Z_{k-1}=O_{k-1}(Z_{k-2}),\ I_{i+1}=O_k(Z_{k-1}).$$

Als ein Beispiel sei hier eine Operation zur Änderung einer Baumstruktur genannt. Diese besteht i. allg. aus zwei Schritten:

1. Änderung der Strukturinformationen,
2. Änderung der eigentlichen Daten.

Ein Systemfehler während der Ausführung einer der Teiloperationen $O_1, \ldots, O_k$ führt somit zu fehlerhaften Daten- bzw. Verzeichnisinhalten. Deshalb ist es sinnvoll, zur Sicherung der Daten in einem PES das Mittel der Transaktion (TA) einzusetzen. Damit wird es möglich, solche oben beschriebenen Operationsfolgen als eine Einheit zu behandeln. Der zeitliche Verlauf einer TA ist in Bild 2 dargestellt.

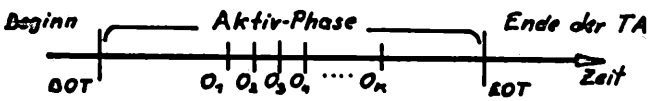


Bild 2: Zeitlicher Ablauf einer Transaktion

Der Umfang einer TA, also die zu ihr gehörenden Operationen, wird in den Nutzerprogrammen durch entsprechende BOT- bzw. EOT-Klauseln definiert. Durch die Routinen des TA-Managments werden folgende Eigenschaften abgesichert:

1. atomar - entweder ist eine TA vollständig ausgeführt oder gar nicht,
2. permanent - die Ergebnisse einer TA bleiben nach EOT unabhängig von Fehlern erhalten.

Außerdem erhält der Nutzer die Möglichkeit, eine TA während ihrer Aktiv-Phase zurückzusetzen, also den Datenzustand vor BOT wieder herzustellen.

4.3. Bereitstellung und Speicherung von Sicherungsinformationen

Jede Maßnahme zur Datensicherung beruht auf der Redundanz von Daten. Auch zur Sicherung der TA-Eigenschaften ist es notwendig, zusätzliche Sicherungsdaten auf einem Log-File zu speichern. Das ist ein sequentieller Datenträger, der für jede TA folgende Informationen enthält:

BOT-Satz	Before-Image	After-Image	...	EOT-Satz

Daten eines Segments
vor nach
dessen Veränderung

Da für alle TA's ein gemeinsames Log-File benutzt wird, sind die Sicherungsdaten der verschiedenen TA gemischt gespeichert. Deshalb enthält jedes Image Identifikationsdaten mit folgendem Aufbau:

Timestamp	TA-no	File-no	Segment-no	Segmentdaten

Das Schreiben der Log-Daten wird vom TA-Managment realisiert. Damit besteht z. B. eine transaktionsorientierte Schreiboperation aus den Schritten:

TAWRITE (TA-no,File-no,Segment-no,Datenadresse)

READ (File-no,Segment-no,Adresse) WRITELOG (TA-no,before,Adresse)	Lesen des alten Segmentinhalts und Schreiben auf Log
WRITE (File-no,Segment-no,Datenadresse) WRITELOG (TA-no,after,Datenadresse)	Schreiben der neuen Daten in File und auf Log

Die Sicherung der TA-Eigenschaften erfolgt durch Auswertung der Log-Daten nach folgendem Schema:

Eigenschaft	Fehlerart	Maßnahmen
atomar	Nutzerfehler	Before-Image der entspr. TA --> DV
	Systemfehler	1. alle unvollend. TA ermitteln 2. Before-Im. d. unvollend. TA --> DV 3. After-Im. d. vollend. TA --> DV
permanent	Plattenfehler	After-Image der vollend. TA --> DV

Das Log-File kann von Zeit zu Zeit reorganisiert werden, da es sonst viele nicht mehr benötigte Daten enthält. Nach einem Crash können die Sicherungsdaten aller unvollendeten TA gelöscht werden. Außerdem können die After-Images der vollendeten TA auf einem Archivlog gespeichert werden, das bei Plattenfehlern gemeinsam mit dem aktuellen Log inspiziert werden muß.

5. Implementation einer Datenbasis für ein PES

Für eine erste Version eines Programmentwicklungssystems /Ko82/ wurde eine Datenverwaltung entsprechend den o. g. Grundsätzen implementiert. Sie ist für den Einsatz im Betriebssystem OS/ES gedacht, kann jedoch bei Bedarf auch auf anderen Rechnern installiert werden. Als Implementationssprache wurde M-PASCAL /BK82/ gewählt, die eine Erweiterung von PASCAL um wesentliche Teile des Modulkonzepts von MODULA-2 darstellt.

5.1. Implementation der Speicheralgorithmen

Es wurde das Filesystem PROFILE /Su84/ realisiert, das auf den im Abschnitt 3 beschriebenen Grundstrukturen beruht. Die vom UNIX bekannten Funktionen, wie CREATE, OPEN, READ, WRITE usw., werden dem Nutzer bzw. Nutzerprogramm als Satz externer Routinen im Sinne von PASCAL zur Verfügung gestellt. Um die Portabilität und die Offenheit des Systems zu gewährleisten, wurde eine Schichtenarchitektur (Bild 3) implementiert.
Soll das System auf einem anderen Rechner installiert werden, so muß der ASSEMBLER-Modul REALACC ausgetauscht bzw. über-

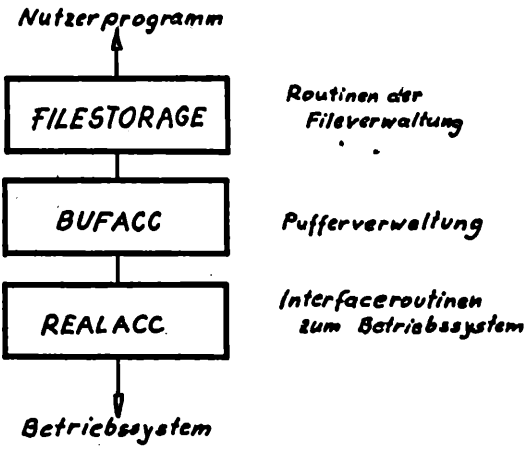


Bild 3 : Schichtenarchitektur von PROFILE

arbeitet werden. Dabei wird lediglich eine Direktzugriffsmöglichkeit auf externe Daten mit eines Blockgröße von 512 Byte gefordert.
Der Modul BUFACC kann ganz entfallen, wenn das aufgrund zu geringen verfügbaren Speicherplatzes erforderlich ist. Dies wird durch gleiche Anschlußbedingungen sowohl für REALACC als auch für BUFACC ermöglicht.

5.2. Implementation des TA-Managments

Der im Abschnitt 4 beschriebene Datensicherungsalgorithmus auf der Basis des Transaktionskonzepts wurde als ein Nutzerprogramm von PROFILE implementiert. Dadurch waren keine Eingriffe in die Filesystemverwaltung erforderlich (Bild 4).
Die PROFILE-Funktionen werden vom Modul TRANSACT verwendet, um das TA-Managment zu realisieren. Der Nutzer hat dann nur noch mit Hilfe von TRANSACT Zugang zu den Filesystemdaten, der ihm durch transaktionsorientierte Operationen ermöglicht

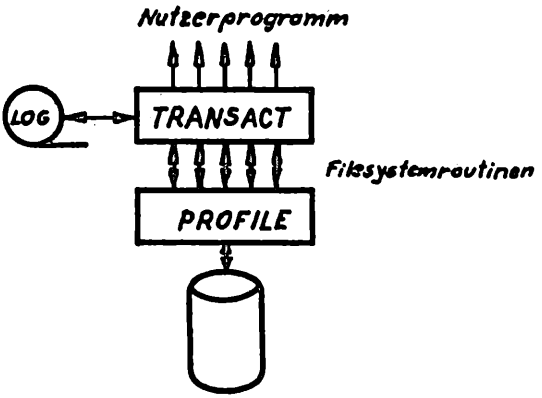


Bild 4: Verbindung Filesystem - TA-Managment

wird. Zusätzlich werden folgende Funktionen angeboten:

TAOPEN - Eröffnen einer Transaktion
TACLOSE - Abschließen einer Transaktion
TAABORT - Abbruch einer Transaktion
RESTART - Wiederanlauf von TRANSACT nach Systemfehler.

Weitere Funktionen zur Datensicherung bei Plattenfehlern sowie zur Reorganisation des Log-Files sind in Vorbereitung.

6. Literatur

/BK82/ Bothe, K., Kosciolowicz, R.: M-PASCAL, Sprachbeschreibung Forschungsbericht Humboldt-Universität zu Berlin, Sekt. Mathematik, 1982

/EH82/ Effelsberger, W., Härder, Th.: Principles of Database Buffer Managment, Forschungsbericht Universität Kaiserslautern, 1982

/Gr78/ Gray, J.N.: Notes on Data Base Operating Systems, in Lecture Notes in Computer Sciences, Vol. 60, Springer-Verlag, 1978

/HHMT84/ Herrlich, O., Horn, E., Müller, D., Tatsch, B.: Softwareentwicklungssysteme - Stand und Tendenzen, in INFO 84, Tagungsberichte, Dresden 1984

/Ho73/ Hoare, C. A. R.: A Structured paging System, The Computer Journal, Vol. 16, No. 3, Aug. 1973

/Ko82/ Kosciolowicz, R.: Konzeption eines Programmierlabors zur Unterstützung des modularen Programmentwurfs mit PASCAL, in bit'82, Tagungsbericht, Berlin 1982

/Ro82/ Rochkind, M. J.: Structure of a Database File System for the UNIX Operating System, The Bell System Technical Journal, Vol. 61, No. 9, Nov. 1982

/RT74/ Ritchie, D. M., Thompson, K.: The UNIX Time-Sharing System, CACM, Vol. 17, No. 7, July 1974

/Su84/ Schuster, H.: Filesystem PROFILE, Version 1.0., Benutzungshinweise, Forschungsbericht, Humboldt-Universität zu Berlin, Sektion Mathematik, 1984

Verfasser:

Dipl.-Math. Harald Schuster
Humboldt-Universität zu Berlin
Sektion Mathematik
1086 Berlin
Postfach 1297

Wagner, Bernd
Strohmeyer, Stephan

Das bildschirmorientierte Simulationssystem TOMAS - Stand der Entwicklung und erste Erfahrungen bei der Anwendung auf Prozesse der Elektronikindustrie

1. Stand der Entwicklung von TOMAS/ES

Das Simulationssystem TOMAS (Technology Oriented Modelling And Simulation) wurde für die Modellierung und Simulation zeit- und zustandsdiskreter technologischer Prozesse für Technologen als Fachnutzer geschaffen. Es ermöglicht die effektive Erarbeitung von Projekt- und Rationalisierungsvarianten solcher Prozesse.
Mit der Entwicklung von TOMAS wurden im wesentlichen folgende drei Grundprobleme gelöst, die eine unmittelbare Anwendung von TOMAS durch Technologen ermöglichen:

1. Ein prozeßorientiertes Modellierungskonzept gestattet die Erfassung und Widerspiegelung komplexer Zusammenhänge eines Realprozesses.
2. Von der methodologischen Gestaltung her wird dem Fachnutzer ein seiner Denkweise entsprechendes Prozeßstrukturierungskonzept in die Hand gegeben, das ihn von Fragen der mathematischen Formalisierung und Algorithmierung prozeßrelevanter Modellbestandteile und deren rechentechnischer Umsetzung (Programmierung) vollständig entlastet.
3. Ein interaktives Zugangskonzept zum System TOMAS auf umgangssprachlichem Niveau unterstützt den Nutzer in seiner fachgebietsbezogenen Denkweise, führt ihn weitgehend bei der Modellierung und ermöglicht Experimente am Modell.

Das grundsätzliche Konzept von TOMAS ist u. a. in /1/, /2/ beschrieben.
Charakteristisch ist, daß dem Nutzer für den Bau des Prozeß-

modells vorgefertigte Bausteine angeboten werden, von denen jeder einen typischen Teilprozeß diskreter technologischer Prozesse algorithmisch nachbildet.
Der Modellierungsablauf gestaltet sich wie folgt:

1. Strukturierung des zu simulierenden Prozesses im Hinblick auf die verfolgten Zielkriterien und unter Beachtung der in den angebotenen Bausteinen fixierten Abläufe

2. Dialogeingabe des im ersten Schritt entstehenden Prozeßmodells in den Phasen
   - Struktureingabe
   - Modellierelemente- (Baustein-) Zuweisung
   - Parameterinitialisierung

Modellmanipulationen in struktureller und funktioneller Sicht sowie Aspekte der Experimentplanung (Ergebnisgewinnung, Zeitsteuerung usw.) sind ebenfalls dialoggestützt einfach angebbar.
Die aktuelle Version TOMAS/ES-1.2 ist noch vorrangig auf die Nachbildung technologischer Basisprozesse der Großserien- und Massenfertigung orientiert, wobei Aspekte der Fertigungssteuerung nur implizit in gewissem Umfang realisiert sind.
Gegenwärtig entsteht die Version TOMAS/ES-2.0, die durch folgende neue wesentliche Merkmale gekennzeichnet ist:

1. Einbeziehung der Fertigungssteuerung (im Sinne eines automatisierten flexiblen Fertigungsabschnitts) in den vorgefertigten Bausteinvorrat

2. Möglichkeit zur automatischen Generierung eines Modellansatzes aus betrieblichen technologischen Daten.

3. weitere Strukturierung des Zugangssystems in methodologischer und funktioneller Hinsicht

Erste Erfahrungen dazu werden zu Beginn des Jahres 1986 vorliegen.

2. Erste Erfahrungen bei der Anwendung von TOMAS

Die Untersuchung komplexer technologischer Prozesse ist infolge ihrer zunehmenden Verflechtung oftmals nur mit Hilfe der Simulation möglich. Der dazu erforderliche hohe Programmieraufwand kann durch die Nutzung geeigneter Simulationssysteme reduziert werden, wobei besonderer Wert darauf gelegt werden sollte, den Technologen zur direkten Nutzung dieser Systeme zu befähigen.

TOMAS bietet aufgrund seines dialogorientierten Konzepts diese Möglichkeit; der Vorrat an modellierenden Elementen (Bausteinen) gestattet die Nachbildung von Großserienfertigungsprozessen am Bildschirm und deren Simulation im Stapelbetrieb. Die Vorteile für den Anwender bestehen - insbesondere gegenüber anderen Simulationssystemen - in der Führung des Nutzers durch den Rechner und der damit verbundenen sofortigen Prüfung der Eingabedaten. Durch eine anwenderfreundliche Gestaltung der Dialogbilder werden fehlerhafte Reaktionen weitgehend vermieden. Das gilt auch für kompliziertere Eingaben, z.B. von empirischen Verteilungsfunktionen, die wertepaarweise bereitgestellt werden müssen.

Modelländerungen sind ebenfalls im Dialog möglich und gestatten damit die Untersuchung von Varianten, wobei sowohl die Struktur des Modells als auch die Eingabedaten variiert werden können. Zusätzlich zur Bildschirmausgabe werden alle für das Modell relevanten Eingaben auf dem Drucker protokolliert, um das Simulationsmodell zu dokumentieren. Die Form dieser Dokumentation kann noch nicht alle Wünsche bezüglich Übersichtlichkeit befriedigen, an einer Verbesserung wird z.Z. gearbeitet. Weitere Forderungen, die sich aus der Arbeit mit dem System in der Erprobungsphase ergaben und z.B. die Modellidentifizierung, die Eingabe der Simulationsdauer u.a. betreffen, wurden bereits realisiert.

In der Erprobungsphase für die Version TOMAS 1.0 wurden Fertigungsprozesse der Elektronikindustrie untersucht. Da für solche komplexe Prozesse in zunehmendem Maße dezentrale Datenverarbeitungsanlagen für die Steuerung eingesetzt werden, stand hierbei nicht die Nachbildung des Materialflusses im Vordergrund, sondern die Simulation des Informationsflusses.

Der Informationsfluß stellt praktisch das Bindeglied zwischen Prozeßanalyse und Prozeßsteuerung dar. Seine Durchdringung ist die Voraussetzung für die Gestaltung eines leistungsfähigen Informationssystems unter Einsatz eines Prozeßrechnersystems.

In /3/ wurde festgestellt, daß sich TOMAS auch für die Modellierung von Informationsflüssen eignet. Die für das Informationssystem erforderlichen Datenmeldungen (z.B. Posteneröffnungsmeldung, Teilschrittmeldung, Postenabschlußmeldung), die im Originalsystem über Terminals eingegeben werden und eine Rechnerelementestruktur zum Zwecke der Verarbeitung durchlaufen, werden in TOMAS durch Operanden nachgebildet und bewegen sich durch eine Struktur von modellierenden Elementen. Dabei werden z.B. mittlere Auslastungen von Terminals und mittlere sowie maximale Längen der davorliegenden Warteschlangen für mehrere Modellvarianten, die sich durch verschiedene Verteilungen für Bearbeitungs- und Reparaturzeiten unterscheiden, ermittelt. Diese Ergebnisse machten es möglich, die bisherige, nur auf Erfahrungen beruhende Einsatzvorbereitung für das Prozeßrechnersystem quantitativ besser zu erfassen und Maßstäbe dafür zu gewinnen, welcher Umfang der Meldungen mit einer bestimmten Gerätekonfiguration noch zufriedenstellend verarbeitbar ist.

Abschließend sei bemerkt, daß die Lösung dieser Problematik mit Hilfe von TOMAS bereits in der Erprobungsphase dieses Simulationssystems erfolgen konnte, daß die Einarbeitungszeit des Anwenders relativ kurz war und daß auch wertvolle Hinweise für die Erhöhung der Nutzerfreundlichkeit von TOMAS gewonnen wurden.

Literatur :

/1/ : Frank, M.: Das interaktive System TOMAS
edv-aspekte 3(1984) H. 2 S. 29 - 32

/2/ : Vavilov, A. A. (Hrsg.) : Modellierung und Simulation von Produktionsprozessen
Berlin 1983 : Verlag Technik S. 179 - 197

/3/ : Engelmann, B.: Einsatzvorbereitung für den Prozeßrechner - Simulation des Rechnereinsatzes in der Fertigung optoelektronischer Bauelemente, Diplomarbeit HUB SE 1985

Dipl.-Ing. S. Strohmeyer, IHS Dresden, Sektion IV
Dipl.-Phys. B. Wagner, HU Berlin, ORZ

Wegel, Günter

DIALOGSYSTEM DIAL FÜR PROGRAMMIERER, TECHNOLOGEN UND ANWENDER

1. Bildschirmgerätesystem EC 7920

Das Gerätesystem EC 7920 ist ein universelles alphanumerisches Kommunikationssystem, das an ESER-Anlagen angeschlossen werden kann. Zum Bildschirmgerätesystem EC 7920 gehören das Bildschirmgerät EC 7927 mit Tastatur und Selektierstift, der Drukker EC 7934, die Lokalsteuereinheit EC 7922 und die Fernsteuereinheit EC 7921. Die Lokalsteuereinheit wird direkt an einen Kanal der Zentraleinheit angeschlossen. Zwischen der Fernsteuereinheit und einem Multiplexsteuergerät (MPD4) wird die Übertragungstechnik für Datenfernverarbeitung geschaltet. Das Multiplexsteuergerät wird an einen Multiplexkanal der Zentraleinheit angeschlossen. An ein Multiplexsteuergerät können auch mehrere Fernsteuereinheiten über jeweils eine Übertragungstechnik angeschlossen werden. Sowohl an einer Lokal- als auch an einer Fernsteuereinheit können bis zu 32 Bildschirmgeräte oder Drucker des Systems EC 7920 in beliebiger Kombination angeschlossen werden.

Der Bildschirm des Gerätes EC 7927 ist in 24 Zeilen und 80 Spalten zur Darstellung jeweils eines alphanumerischen Zeichens eingeteilt. Im allgemeinen wird vom Programm ein Bild vorgegeben, welches der Bediener mittels Tastatur verändert. Das vom Programm vorgegebene Bild kann formatisiert sein. Das heißt, es kann in Felder mit folgenden Eingenschaften eingeteilt sein:

- ungeschützt oder geschützt,
- numerisch oder alphanumerisch,
- normal hell anzeigbar, intensiv anzeigbar oder nicht anzeigbar und
- mit Selektierstift selektierbar oder nicht.

Dieses Feldkonzept der Gerätetechnik bietet die Möglichkeit, anwenderfreundliche Dialogprogramme zu entwickeln. Das Dialogsystem DIAL stützt sich auf dieses Feldkonzept und ist somit für eine andere Gerätetechnik nicht nutzbar.

2. Komponenten von DIAL

PORG: Diese Komponente ist die dialogfähige Variante des Programmsystems PORG zur Verwaltung und Pflege von Programmdaten und enthält Funktionen zum:

- Darstellen, Zufügen und Modifizieren von Quelltexten
- Darstellen des Inhaltsverzeichnisses von Bibliotheken und des Operativspeichers
- Löschen von Bibliotheksbeständen bzw. Tabellen
- Aufruf von (Stapel-)Operationen des Programmsystems PORG wie MODIFY, COPY, LIST, PREPARE usw.

LIST: Mit dieser Komponente besteht die Möglichkeit, Drucklisten darzustellen und Zeilen der Druckliste zu modifizieren. Die Drucklisten müssen sich hierzu in einem speziell organisierten Tabellenspeicher befinden. Für spezielle Drucklisten bzw. spezielle Anwendungsfälle können geeignete Formate zur Darstellung der Drucklisten generiert werden. Neben der bereits erwähnten Funktion enthält die Komponente noch weitere Funktionen zum:

- Laden von Drucklisten in den Tabellenspeicher
- Zurückspeichern von Drucklisten aus dem Tabellenspeicher
- Ausgabe von Drucklisten auf Schnelldrucker
- Ausgabe von Drucklisten auf einem EC 7920-Drucker
- Darstellen des Inhaltsverzeichnisses des Tabellenspeichers
- Löschen von Drucklisten aus dem Tabellenspeicher.

EXEC: Hiermit wird der Aufruf und die Abarbeitung von Programmen gesteuert, die im Stapelbetrieb als Jobstep laufen. Durch eine geeignete Generierung kann der Aufruf spezieller Programme bzw. Programmklassen (z. B. Assembler, Programmverbinder, Dienstprogramme, bestimmte Anwenderprogramme) besonders unterstützt werden.

JOB: Der Datenendplatznutzer kann mit dieser Komponente Jobs einschließlich Jobbeschreibung auf einer im Rechenzentrum zentral angelegten Jobsammeldatei abspeichern, d. h. an den Arbeitsvorbereiter absenden. Der Arbeitsvorbereiter verwaltet diese Jobsammeldatei und veranlaßt die Abarbeitung der auf dieser Datei eingegangenen Jobs. Der Arbeitsvorbereiter bestimmt hierbei die Reihenfolge und die EDVA für die Abarbeitung der Jobs.

AV: Diese Komponente ist das Hilfsmittel für den Arbeitsvorbereiter zur Verwaltung der Jobsammeldatei, für das Umsetzen der Jobs in die Warteschlangen der EDVA und zur Steuerung von Wiederholläufen. In die Jobsammeldatei werden auch generierte Jobs der Stapelverarbeitung für Massendatenverarbeitung eingespeichert und somit zur Abarbeitung durch den Arbeitsvorbereiter mit Hilfe dieser Komponente veranlaßt.

BATCH: Die Anwenderprogramme (Operationen), die im Stapelbetrieb unter dem einheitlichen Steuerrahmen BATCH laufen, können im Dialogbetrieb mit Hilfe der Komponente BATCH aufgerufen und abgearbeitet werden. Desweiteren können dialogführende Anwenderoperationen aufgerufen werden, wie z. B.: zur Darstellung und Modifizierung von Referenzspeicherdaten von Anwendungsfällen und zur Darstellung und Modifizierung von direktgespeicherten Massendaten.

USER: Diese Komponente ist eine Erweiterung der Komponente BATCH, die folgende Möglichkeiten zusätzlich enthält:

- Steuerung der Auflösung und Abarbeitung mehrstufiger Makros, die jeweils eine Folge von Anwenderoperationen repräsentieren
- Steuerung von vorbereiteten Dialogabläufen (Modifizierung und Abarbeitung einer Folge von Makros)
- Generierung von Stapeljobs der Massendatenverarbeitung

3. Nutzung von Datenendplätzen

Die Komponente JOB wird von allen Datenendplätzen (DEP) genutzt.

Programmierer: Der DEP "Programmierer" besteht vor allem in der Nutzung des TSO-Kommandos TEST (zur Testung von Assemblerprogrammen), der DIAL-Komponenten PORG (zur Programmdatenpflege), EXEC (zum Aufruf des Programmverbinders, des Assemblers und des Superzaps-Programms) und LIST (zum Darstellen und Drucken von Protokollen) sowie des TSO-Kommandos SUBMIT (zum Absetzen von Hintergrundjobs).

Projektant: Vom Projektant werden vor allem die DIAL-Komponenten PORG (zur Projektdatenpflege), BATCH und USER (zum Testen von Projekten und zum Einrichten und Erproben von Anwendungsfällen) und LIST (zum Darstellen und Drucken von Protokollen

sowie zum Darstellen (Überprüfen) der vom Projekt erzeugten Ergebnislisten) sowie die TSO-Funktion SUBMIT.

Arbeitsvorbereiter: Der DEP "Arbeitsvorbereiter" besteht in der Nutzung der Komponente AV. Außerhalb der TSO-Betriebszeit wird DIAL im BTAM-Betrieb genutzt. Für den Ausfall der Bildschirmtechnik gibt es eine Stapelversion zur Verwaltung der Jobsammeldatei.

Durchlaufbetreuer: Zur Bearbeitung von Aufträgen der Massendatenverarbeitung werden Komponenten USER (zur Jobgenerierung), EXEC (zum Aufruf von Dienstprogrammen und zur Jobgenerierung) und LIST (zum Darstellen und Drucken von Protokollen) genutzt.

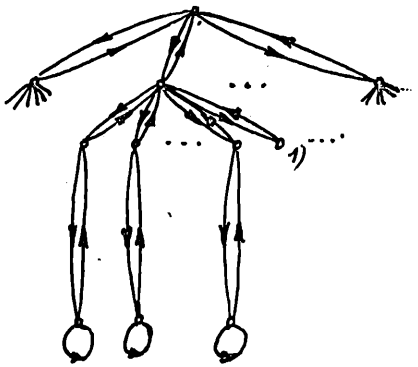
Anwender: Die Aufgaben der einzelnen Anwender sind sehr unterschiedlich und, auch die Anwendung der EDV ist sehr unterschiedlich. Sie reicht von der Nutzung eines einzigen relativ starren Projektes bis hin zur Nutzung von mehreren komplexen und flexiblen Projekten. Somit ist auch die Nutzung von DIAL bei den Anwendern weitaus unterschiedlicher als bei den bisher beschriebenen DEP. Vom Anwender werden die Komponenten USER, BATCH, EXEC, PORG und LIST für folgende Tätigkeiten genutzt:

- Eingabe und Pflege von Listmusterabrufen zur Erzeugung von Plandokumenten und anderen Ergebnislisten
- Pflege von Nomenklaturen im Referenzspeicher
- Pflege von Formelpaketen im Referenzspeicher
- Datenpflege (bei geringem Umfang)
- Durchführung von Rechnungen (bei geringem Umfang)
- Erzeugung von Drucklisten (bei geringem Umfang)
- Darstellung von Drucklisten mit statistischen Daten
- Überprüfung der Ergebnislisten von Modellberechnungen
- Korrektur von Plandokumenten und anderen Ergebnislisten

4. Prinzip der Steuerung und der Dialogführung von DIAL

Das Dialogsystem DIAL besteht aus dem Steuerrahmen und den vorgestellten Komponenten. Im Steuerrahmen und in allen Komponenten von DIAL ist ein einheitliches Prinzip der Dialogführung und der Bildgestaltung realisiert. Den Dialog kann man als Struktur interpretieren, wobei die einzelnen Punkte ein Bild repräsentieren. In Abbildung 1 ist ein sehr vereinfachtes Schema der Dialogstruktur dargestellt.

Abbildung 1:



Auswahl der Komponente

Auswahl der Funktionen

Auswahl der Daten, Auswahl bzw. Eingabe von Parametern und bei nicht dialogführenden Funktionen 1)
Ausführung der Funktionen

Datendialog (Darstellung und/ oder Modifizierung von Daten)

Die Tasten zur Achtungsunterbrechung finden hierbei weitestgehend einheitliche Verwendung.
Die Nutzung von DIAL ist in den folgenden drei Betriebsarten möglich:

- Simulations-Betrieb: zur Entwicklung bzw. Weiterentwicklung von DIAL ohne Inanspruchnahme der Gerätetechnik EC 7920
- BTAM-Betrieb: für den Einsatz weniger Bildschirmgeräte (max. 5) im Nahanschluß geeignet
- TSO-Betrieb: für den Einsatz einer größeren Anzahl von Bildschirmgeräten und für den Einsatz von fernangeschlossenen Bildschirmgeräten geeignet. Für diese Betriebsart ist der Einsatz des Teilnehmersystems TSO eine Voraussetzung.

5. Möglichkeiten und Hilfsmittel zur Entwicklung bzw. Weiterentwicklung von DIAL

Zunächst gibt es die eigens für DIAL entwickelte Zugriffsmethode QDIS. Mit QDIS wird neben den Zugriffen zum Bildschirmgerät, die Pufferverwaltung, die Umrechnung der Bildadressen und die Analyse der Eingabepuffer unterstützt. Damit entfallen für den Anwendungsprogrammierer die immer wiederkehrenden Routinearbeiten. Durch die Nutzung von QDIS kann man auch unabhängig von der Betriebsart (s. Punkt 4.) programmieren.

Weiterhin gibt es das Hilfsmittel DIAL-H. Das Hilfsmittel DIAL-H ist darauf gerichtet, daß eine einheitliche Gestaltung des Dialogs (Dialogstruktur, einheitliche Tastenbelegung usw.) gewährleistet ist.
Während QDIS eine Zugriffsmethode ist - einem READ-WRITE-Niveau entspricht - entspricht DIAL-H dem Charakter nach einem Programmiersystem. Jedoch unterstützt DIAL-H nur die Arbeit mit dem Bildschirm und nicht die Arbeit mit irgendwelchen Speichern. Somit müssen die Komponenten von DIAL ihre Speicherarbeit zwar selbst organisieren, aber DIAL-H ist universell einsetzbar.

<u>Literatur:</u>

Dokumentation des Bildschirmsystems EC 7920, Robotron
Dokumentation des Teilnehmersystems TSO, Robotron
Dokumentation von UNI-3, Robotron
Dokumentation von DIAL, SPK/RZ

Winkler, F. / Horn, E.

Ein Beitrag zur Modularisierung beim Softwareentwurf

1. Einleitung

Ein wichtiges Prinzip bei der Softwareentwicklung ist die Modularisierung geworden.
Wie in den Technikwissenschaften, so bringt auch in der Softwaretechnologie die Arbeit mit Moduln viele Vorteile für die Realisierung und Nutzung komlexer Systeme.
Entsprechende Konzepte zum Aufstellen von modularer Software werden in modernen Programmiersprachen weitestgehend berücksichtigt (z.B. MODULA2, ADA).
Es besteht somit die Aufgabe, für ein zu realisierendes Softwareprodukt einen modularen Programmentwurf zu erstellen. Die Festlegung der modularen Programmstruktur eines Softwareprodukts muß bereits während des Softwareentwurfs mit den zur Verfügung stehenden Informationen erfolgen.
In der Literatur sind ausführlich das Aufstellen eines Systementwurfs und die Anforderungen an eine modulare Beschreibung dokumentiert (z.B. in /KIMM79/, /SCHN79/). Wie ausgehend von einem Systementwurf eine modulare Programmstruktur erstellt werden kann, ist aber in der Literatur nur wenig ausgeführt.
Arbeiten zur Analyse des Softwaresystementwurfs (vgl. /HORN85/) zeigen, daß ein Systementwurf unabhängig von einer gewählten Softwareentwurfsmethode und der damit verbundenen Form der externen Beschreibung als eine Struktur von Aktionen und Datenelementen mit Beziehungen zwischen diesen dargestellt werden kann.
Der Prozeß der Modularisierung ist eine Transformation der Struktur des Systementwurfs in die Struktur des Programmentwurfs. Sie erfolgt in der Regel dadurch, daß eine Abgrenzung und bzw. oder eine Zusammenfassung von Elementen zu Teilsystemen des Systementwurfs nach bestimmten Kriterien vorgenommen wird.

Diese sind von speziellen Nutzeranforderungen und der konkreten Basismaschine abhängig.
Nachfolgend soll ein Ansatz vorgestellt werden, wie diese Transformation realisiert werden kann.
Entsprechend der erläuterten Strategie kann ein rechnergestütztes Werkzeug zur Verfügung gestellt werden, das bestimmte Arbeiten bei der Ableitung von Moduln unterstützt.

2. Ableitung von Moduln aus einem Systementwurf

2.1. Darstellung des Systementwurfs

Der Systementwurf kann als ein formales System beschrieben werden:

$$SE = (ES, R_{SE})$$

Die Systemelemente ES werden durch die Mengen der Systemaktionen und Datenelemente bestimmt:

$$ES = \{A, D\}$$

Die Beziehungen zwischen den Elementen werden durch R_{ES} angegeben:

$$R_{SE} = \{R_{SE1}, R_{SE2}, R_{SE3}\}$$

$R_{SE1} \subseteq D \times A$ - Aktion a_i benötigt Datenelement d_j

$R_{SE2} \subseteq A \times D$ - Aktion a_i erzeugt Datenelement d_j

$R_{SE3} \subseteq A \times A$ - Aktion a_j folgt auf Aktion a_i

Durch Zusammenfassen von Elementen des Systementwurfs lassen sich Teilsysteme abgrenzen:

$$TS = (ETS, R_{TS})$$

Die Auswahl der Elemente und der Beziehungen zwischen diesen für das betreffende Teilsystem wird dabei durch ein Prädikat bestimmt:

$$ETS = \{A_{TS}, D_{TS} \mid P_{TS}\}$$

$$R_{TS} = \{R_{i_{TS}} \mid P_{TS}\}$$

Der Systementwurf kann somit auch durch eine Struktur von Teil-

systemen und den Beziehungen zwischen diesen angegeben werden:

$$TSS=(TS, R_{TSS})$$

Die Beziehungen zwischen den Teilsystemen (R_{TSS}) sind eine Menge von Relationen, die bestimmt sind durch die Relationen zwischen den Elementen, die verschiedenen Teilsystemen angehören:

$$R_{TSS}=\bigcup_{i=1}^{n} (R_{ES_i} \mid ets_{TS_k} \in VB(R_{ES_i}) \wedge ets_{TS_l} \in NB(R_{ES_i}))$$

2.2. Darstellung der modularen Programmstruktur

Die modulare Programmstruktur kann analog dem Systementwurf als ein formales System dargestellt werden:

$$PS=(EPS, R_{PS})$$

Als Elemente dieses Systems treten Modulbeschreibungen und die Beschreibung der Umwelt auf:

$$EPS=\{M_B, U_B\}$$

Für die Darstellung der Modulbeschreibungen wird die Moduldefinition nach /BALZ78/ zu Grunde gelegt:
"Ein Modul ist ein Untersystem, das folgende Eigenschaften besitzt:

1. Bereitstellung einer Funktion oder einer Datenabstraktion;
2. Kontextunabhängigkeit
   Die Kontextunabhängigkeit beinhaltet, daß der Modul von der Modulumgebung unabhängig entwickelbar, übersetzbar, testbar bzw. verifizierbar, wartbar und verständlich ist.
3. Spezifizierung durch Schnittstellenbeschreibung;
4. Niedrige Schnittstellenkomplexität."

Die Schnittstellenbeschreibung entspricht der genannten Modulbeschreibung. Diese enthält alle relevanten Informationen über den Modul, die der Modulumgebung bekannt sein müssen. So gehören dazu unbedingt die Angaben, welche Ressourcen der Modulumgebung bereitgestellt werden und welche Ressourcen von der Modulum-

gebung benötigt werden. Eine Modulbeschreibung kann somit als ein Tupel von Informationen über einen Modul betrachtet werden.
Entsprechend der Moduldefinition treten Moduln vom Typ "Funktion" und "Abstrakte Datenstruktur" auf. Eine ausführliche Darstellung der Modultypen ist in /WINK85/ enthalten. Eine analoge Beschreibungsform gilt für die Umwelt.

Die Beziehungen zwischen den Elementen der Programmstruktur werden beschrieben durch:

$$R_{PS} = \{R_{PS1}, R_{PS2}\}$$

$R_{PS1} \subseteq M_B \times M_B$ - Modul m_i benutzt Modul m_j

$R_{PS2} \subseteq M_B \times U_B$ - Modul m_i steht mit einem Umweltelement u_j in Beziehung

Neben diesen Beziehungen zwischen den Elementen besteht die Forderung nach einer baumartigen Struktur. Da keine weiteren Einschränkungen über die Nutzungsrelation bestehen, kann die modulare Programmstruktur als eine irreflexive Halbordnungsrelation betrachtet werden. Durch Klassenbildung der Moduln können Hierarchieebenen eingeführt werden.

2.3. Modularisierungsstrategie

Zur Herleitung von Moduln müssen die verschiedensten Kriterien entsprechend der Moduldefinition sowie der Basismaschine beachtet werden. Dazu zählt:

- Realisierung von datenstrukturunabhängigen Moduln,
- Erzielung einer Kontextunabhängigkeit durch eine hohe innere Festigkeit und geringen Datenaustausch zwischen den Moduln,
- Aufbau von Moduln mit abgegrenztem Funktionsumfang und entsprechender minimalen Schnittstellenkomplexität,
- Beachtung statischer und dynamischer Realisierungseigenschaften wie Modulgröße, Speicherbedarf, Laufzeit, Aufrufhäufigkeiten usw.

Je nach Menge der vorliegenden Informationen kann ein nutzer-

spezifischer Programmentwurf erstellt werden. Dabei können, ausgehend von den verwendeten Informationen und der Priorität der genannten Kriterien, verschiedene Modularisierungsstrategien angewendet werden. Mittels einer entsprechenden Bewertung des erstellten Programmentwurfs kann die für den Nutzer optimale Variante ermittelt werden.
Bei dem Aufstellen einer modularen Programmstruktur auf der Grundlage der mit dem Systementwurf bereitgestellten Informationen kann nur ein Teil der wichtigen Kriterien beachtet werden. Angaben die die Realisierung des Moduls betreffen, liegen nicht vor. Da erst die Moduln durch Abspalten von Teilsystemen aufgestellt werden sollen, können auch noch keine Aussagen über Beziehungen zwischen den Moduln getroffen werden. Diese ergeben sich erst mit der Formulierung der Moduln.
Es wird davon ausgegangen, daß im Vordergrund die Forderung nach logisch in sich abgeschlossenen Moduln zur Erzielung einer Kontextunabhängigkeit steht.
Aus dem Systementwurf sind die Beziehungen zwischen den Systemelementen ersichtlich. Es können deshalb solche Systemteile zu Teilsystemen zusammengefaßt werden, die enge Beziehungen besitzen und somit als logisch abgeschlossene Einheit eine hohe innere Festigkeit aufweisen. Gleichzeitig ergibt sich daraus, daß die Beziehungen zwischen den so entstandenen Moduln auf ein Minimum reduziert werden. Werden die abgegrenzten Teilsysteme als Moduln mit einem "gültigen" Modultyp aufgestellt, wird weiterhin die Datenstrukturunabhängigkeit der Moduln erreicht.
Grundlage der Bestimmung der Teilsysteme ist die Formulierung eines Prädikats, anhand dessen notwendige Beziehungen zwischen den Elementen gefordert werden.
Zur Darstellung der inneren Festigkeit sind in /BALZ82/ verschiedene Bindungsarten beschrieben. Je nach Stärke dieser Bindungen werden folgende Bindungsarten unterschieden, deren Festigkeit in der Reihenfolge der Aufzählung zunimmt:
- zufällige Bindung
- logische Bindung
- prozedurale Bindung

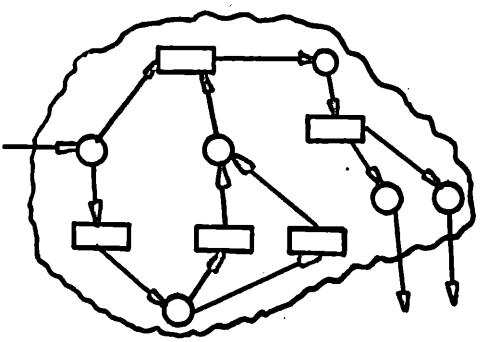
- kommunikative Bindung
- sequentielle Bindung
- informale Bindung
- funktionale Bindung.

Zur Erläuterung der Bindungsarten und für die Bestimmung der Prädikate für die entsprechenden Teilsysteme sei auf /WINK85/ verwiesen.

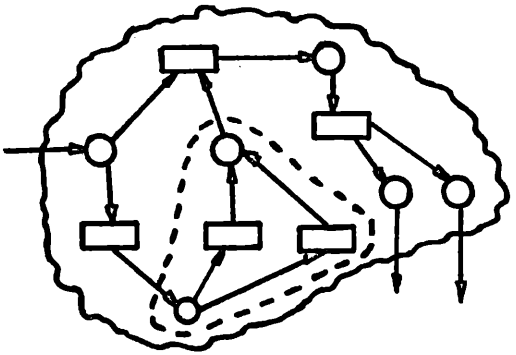
Untersuchungen zeigten, daß nur die vier stärksten Bindungen für die sinnvolle Bildung von Teilsystemen in Frage kommen. Die restlichen schwächeren Bindungsarten garantieren nicht die logisch abgeschlossene Einheit der entstehenden Moduln, da nur wenig Beziehungen zwischen den Systemelementen bestehen würden. Eine Ausnahme bildet die zufällige Bindung. Für alle Elemente des Systementwurfs, die nach erfolgter Abgrenzung von Teilsystemen wegen zu schwachen Beziehungen zu anderen Systemelementen noch nicht mit erfaßt wurden, können zufällige Beziehungen zur Bildung der Moduln benutzt werden.

Folgendes Beispiel soll das Prinzip der Modularisierung zeigen. Zur Darstellung der Aktionen werden Kreise und für die Datenelemente Rechtecke verwendet.

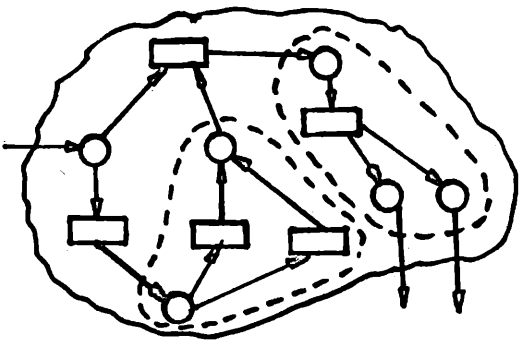
Als Ausgangsstruktur liegt die Struktur des Systementwurfs vor (1). Auf diese Struktur lassen sich notwendige Beziehungen für zwei Bindungsarten anwenden. Erstens eine sequentielle Bindung (2) und zweitens eine kommunikative Bindung (3). Zum Schluß kann aus den verbleibenden Elementen noch ein Teilsystem gebildet werden (4).



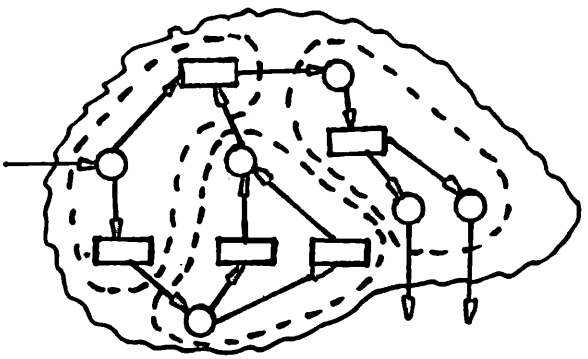
(1)



(2)



(3)



(4)

Neben dem Abgrenzen von Teilsystemen auf der Grundlage der ausgewählten Bindungen sind weitere Forderungen zu beachten. Zum Aufstellen der beschriebenen modularen Programmstruktur muß die geforderte Aufrufrelation eingehalten werden. Dies bedeutet, daß die Beziehungen der Teilsysteme untereinander beachtet werden müssen, damit zwischen den aufgestellten Moduln eine hierarchische Aufrufrelation entsteht. Dazu gehört die Festlegung der Datenelemente, die der Kommunikation zwischen den Teilsystemen dienen und die entsprechende Zuordnung zu einem Teilsystem. Es bedarf außerdem der Formulierung eines Topmoduls, der alle von der Umwelt ausgehenden Systemaktivierungen realisiert.
Da die zur Modularisierung benutzten Bindungsarten unterschiedliche innere Festigkeiten der Moduln gewährleisten, sollte immer die maximal mögliche Anzahl von Teilsystemen der jeweils möglichen stärksten Bindung abgegrenzt werden.
Ein Algorithmus für die Modularisierung läßt sich so formulieren, daß zyklisch Teilsystem für Teilsystem, immer mit höchstmöglicher Bindung, abgegrenzt wird. Die Abgrenzung geschieht in der Form, daß die die Bindung realisierenden Elemente und entsprechende Parameter zur Teilsystemumgebung aus dem Systementwurf gestrichen werden. Gleichzeitig wird mit ihnen eine Modulbeschreibung als Element des modularen Entwurfs aufgebaut. Zum Abschluß muß zusätzlich eine Beschreibung eines Topmoduls aufgestellt werden.
Das Streichen der Elemente eines Teilsystems im Systementwurf bedeutet dabei:

- Bildung der Differenzmenge zwischen den im System enthaltenen Elementen und den im betreffenden Teilsystem sowie als Parameter berücksichtigten Elementen.
- Eintragen entsprechender Vermerke in den Systementwurf über die Elemente, die bereits in Teilsystemen angelegt sind, aber noch weitere Beziehungen zu anderen Systemelementen aufweisen.

Somit ist gewährleistet, daß nach jedem Abgrenzen eines Teilsystems ein aktueller Systementwurf vorliegt.

Die im Ergebnis der Modularisierung vorliegende Struktur kann nach den verschiedensten Gesichtspunkten bewertet werden. So z.B. nach:

- durchschnittliche Datenkopplung zwischen den Moduln,
- Anzahl der Moduln mit zufälliger Bindung,
- durchschnittliche Schnittstellenbreite,
- Angaben zur Metrik der Struktur.

Je nach den erhaltenen Bewertungsaussagen kann die modulare Struktur modifiziert werden bzw. kann die Modularisierung in einer anderen Reihenfolge der Anwendung der Bindungsarten vorgenommen werden.

3. Werkzeug zur Modularisierung

Innerhalb der Arbeiten zur automatengestützten Programmentwicklung soll ein Werkzeug geschaffen werden, welches die vorgestellte Modularisierungsstrategie unterstützt.

Dieses Werkzeug soll folgenden Funktionsumfang umfassen:

- Funktion zum Erfassen des Systementwurfs,
- Anzeigefunktionen für
  + Aktionen, Datenelemente und Relationen zwischen diesen Elementen,
  + Daten- und Steuerfluß des Systems,
  + im System erkennbare Beziehungen der Bindungsarten,
- Modularisierungsfunktion des beschriebenen Algorithmus,
- Funktionen zur gezielten Auswahl von Teilsystemen,
- Anzeigefunktion für die Elemente der modularen Struktur,
- Bewertungsfunktionen für die modulare Struktur,
- Manipulationsfunktionen für die modulare Struktur und
- Analysefunktionen des Export-/Importverhaltens der Moduln.

Mit diesem Funktionsumfang kann die Ausgangsstruktur des Systementwurfs und die erstellte modulare Struktur so lange manipuliert werden, bis eine den Nutzerwünschen angepaßte Zielstruktur vorliegt.

Für die beiden mit dem Werkzeug arbeitenden Strukturen werden externe Sichten vorgeschlagen, die eine effektive nutzerfreundliche Arbeit sichern. Besonders die erstellten Modulbeschrei-

bungen müssen so aufgebaut sein, daß man sie in weiteren Softwareentwicklungsschritten anwenden kann. So bestehen z.B. Anschlußbedingungen, daß die Modulbeschreibungen in eine modulare Struktur einer Programmiersprache überführt werden.

Literatur

/BALZ78/ Balzert, H.
Vergleichende Betrachtungen modularer Sprachkonzepte
Springer-Verlag Berlin Heidelberg New York, 1978

/BALZ82/ Balzert, H.
Die Entwicklung von Software-Systemen
Bibliographisches Institut AG, Zürich 1982

/HORN85/ Ein Ansatz zur Analyse und Synthese von Softwarelebenszyklen, Forschungsbericht
TU Dresden, Sektion Informationsverarbeitung 1985

/KIMM79/ Kimm, R., Koch, W., Simonsmeier, W., Tontsch, F.
Einführung in Software Engineering
Walter de Gruyter, Berlin New York 1979

/SCHN79/ Schnupp, P., Floyd, C.
Programmentwicklung und Projektorganisation
Walter de Gruyter, Berlin New York 1979

/WINK85/ Modularisierung beim Softwareentwurf
Forschungsbericht
TU Dresden, Sektion Informationsverarbeitung 1985

Dipl.-Ing. Frank Winkler

Doz. Dr.-Ing. Erika Horn

TU Dresden
Sektion Informationsverarbeitung
8027 Dresden
Mommsenstr. 13

COMPUTER AIDED MANAGEMENT SYSTEM AT "DELEP" CONSTRUCTION AND CIVIL ENGINEERING COMPANY

János-Péter Zilahy, Deputy General Manager, "DELEP" Construction and Civil Engineering Company, H-6701 Szeged

Summary

This paper presents the main activities and organisational structure of the company. It gives a brief historical outline on the development of using computer systems in the last 20 years at the company. It refers further on to the nowadays implementations in the management as well as in the production scheduling and control areas. The author tries to make a short forcast for the near future's ways of further development in this field.

1. An Outline Profile

DELEP is one of the largest construction companies in Hungary. Its annual gross production value amounts to more than 4 billion Fts. There are 6300 persons employed, a total number of staff and workers.

The annual business turnover of the company is realized in the following areas of operation:

- construction activities 62 % /of which 50 % is housing projects/
- civil engineering and structural erections 8 %
- reinforced concrete, joinery products and steel fabrications 30 % /of which 20 % is sold commercially/
- technical design and project preparation services 1 %
- construction abroad 2 %

The company has multi-profile subsidiaries in Budapest and Békéscsaba /with a joint staff of 1400 persons/. The mother company is located in Szeged. It has 3 industrial plants, training facilities and a hostel for workers as a number of administrative buildings.

The construction activities of the company are based on standard technologies, some of which are protected by patent. The machine fleet and the production units have been developed according to these standard technologies.

The management system of the company operates at two basic levels: /The schematic drawing of the organization structure is shown in Fig.1./.

a/ Business units

Project preparation, project control, supervision and accountancy are perfomed by a number of financially independent /so called self-accounting/ executive organisations with a total manpower of 4700 people. Here follows a list of these organisations:

- Technical Design Office
- Foreign Trade Office
- Management of Installation and Finishing Works
- Management of Building Plants
- Management of Civil Engineering Plant and Public Works
- Management of Prefabrication Plant /Housing Factory/
- Supply Management

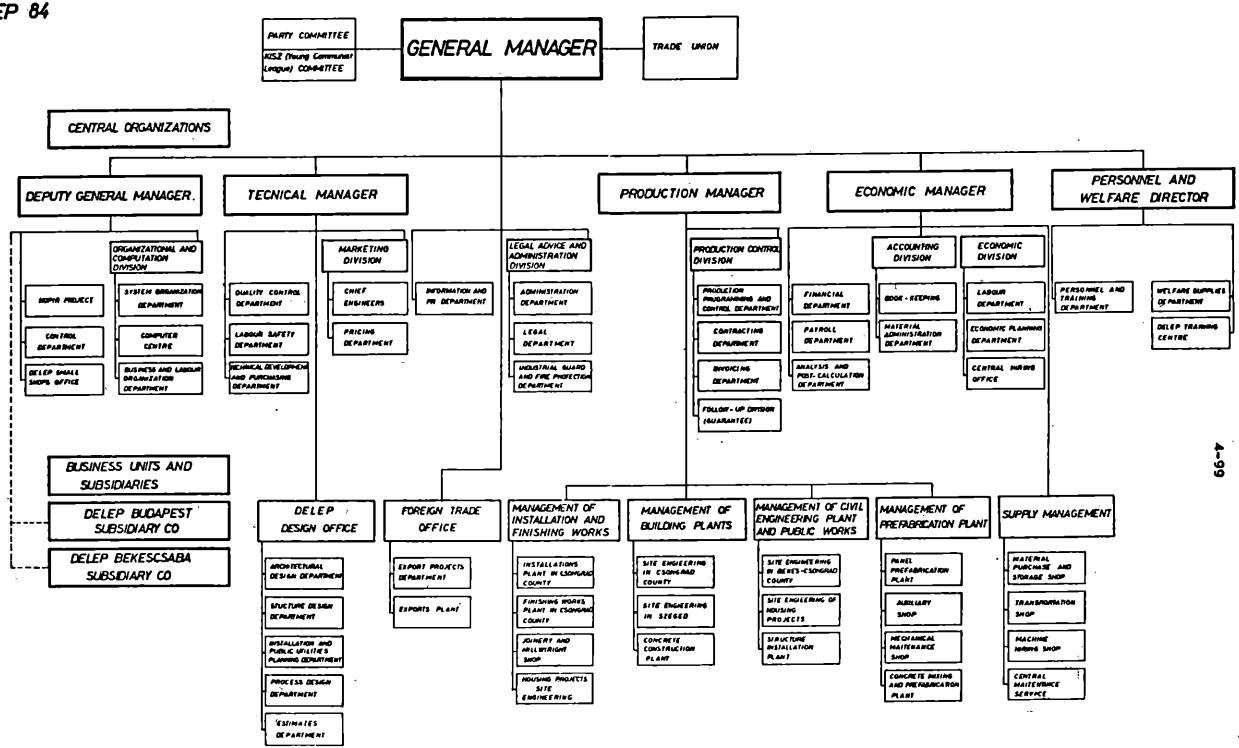
b/ Central organisations

Above this level there is a central management organisation with a staff of 400 people who are responsible for marketing administration, technical development and innovation.

The managing director has one deputy and a number of branch directors /technical, production, business, personnel/. The directors control the activities of divisions and departments of their own branch organisations.

DELEP 84

FIGURE NO. 1



The computer technology division works under the direct supervision of the deputy managing director /staff: 50 people/. The technical configuration available at the present time is described in appendix 1.

2. The application of computer technology in the management activities of the company

The company has been using computerized data processing techniques in its management activities since 1964. The objectives of employing such techniques were in each case determined by the current management problems of the company as well as by the availability of appropriate equipment and trained personnel.

a/ By the mid-60's, computerized data processing was mainly used in the accounting and book-keeping procedures. At that time a batch system was used and processing was done off-site at a computer services centre on a BULL-GE GAMMA 115 machine.

b/ By the end of the 60's, the large scale housing projects and the growing industrial base of the company allowed for the application of a modified work distribution system along the lines of technolgical specialization.

Based on a long-term agreement with the Institute for Building Economy and Organization /ÉGSZI/ we started to implement new organization and management methods at our company /network planning, linear production control, etc./. To support such new management techniques, certain areas of on-site and central production control received computerized support. The KOMPLETER software package helped us in establishing the time and manpower requirements of our construction projects, and helped us in developing time-tables and program schedules for each project, including capacity calculations and capacity utilization analysis for the several executive organizations. The analysis and

the evaluation of the information provided by the computer took place at the central computer application department /Programming Department/ of our company.

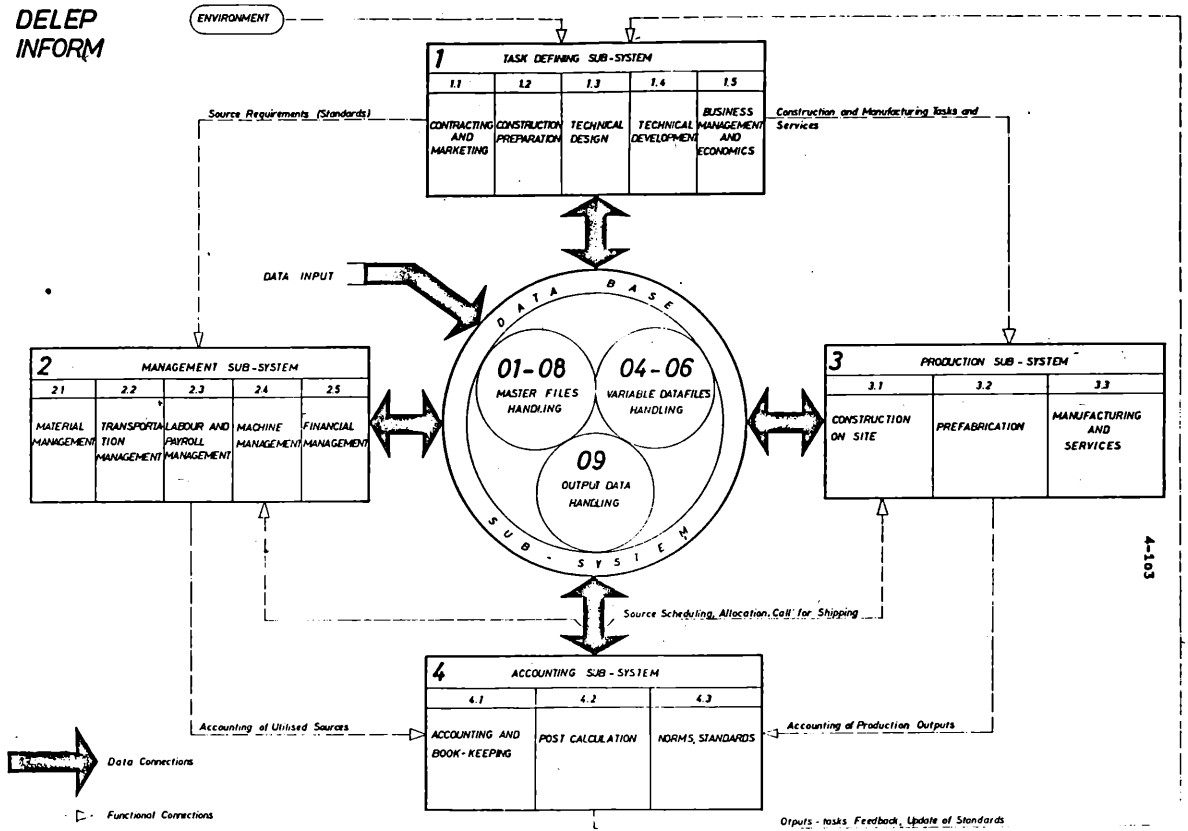
c/ In 1971, production was started at our concrete pre-fabrication plant /with a nominal capacity of 2500 housing units per year/. From the very start, we considered it important to co-ordinate and optimize the fabrication, transportation and production activities at this plant. By this time, the József Attila University at Szeged /JATE/ had already started to train computer scientists and the Hungarian institute for computer technology SZÁMOK ran courses for system designers. We employed a number of these graduated people and by the same time 30 persons from the staff were trained for using computers in a postgraduate course.
These developments allowed us to develop our own control programs for concrete panel production.
The system designed by our own system designers was based on the MINSZK-22 machine of the Szeged University. With the aid of this software, it was possible to determine the requirements for the means of production for a given year from the annual construction schedules. It was also possible to determine the utilization of the concrete casting moulds /dies/.

d/ As the sales increased, the company expanded its capacitiy and in 1975, it gradually started to change its former one-level management system, developing a new two-level system and creating on the second level a number of independent operation managements.
To follow-up these changes, the system designers now had to develop a computer aided management and control system which would define the tasks for each of these financially independent /self-accounting/ executive divisions and

provide adequate information and evaluation capabilities concerning the particular technological processes. Our company has established a computer centre in a joint effort with ÉGSZI, based on ES 1020 type Bulgarian computers /CPU 256 Kbyte/. The conceptual diagram of the system developed on this background is shown in Fig.2. The system called "DÉLÉP INFORM" has been continuously expanded and developed since that time. To speed up implementation, the company rents a number of program packages and it has only developed those sub-systems in its own realm, which were not available in the market either for purchase or for renting; or sub-systems, which included certain elements not covered by commercially available packages. The various sub-systems are supported by common data stores but the size and capacity of the mainframe computer do not allow for the development and implementation of data base manipulating systems. The braekdown systems used for the common masterfiles are described in appendix 2.

e/ During the last few years, the decline in the volume of state sponsored housing developments and the increasingly strong role of privately commissioned dwelling houses in the construction market necessitated further structural changes in our operations. First of all, we had to broaden the range and assortment of our pre-fabricated products. Because of this, production control became more complicated and the new requirements could not be met by the software package previously used. At a competition sponsored by the National Committee for Technical Development /OMFB/, the Central Bureau of Statistics /KSH/ and the Ministry for Construction and Urban Development our company won a grant for the development of an "Operative Control System for Pre-fabricated Panel Based Housing Construction" - HOPIR. In 1982, an ES 1011 type /CPU 1 Mbyte/ machine

Figure No.2



COMPUTER AIDED INFORMATION SYSTEM OF DELEP

Ing. J.P. Zádný 1984.

was put into operation at our pre-fabrication plant.
The terminals of the system were installed at the various workshops and transportation check points. The terminals are operated by foremen.
The logical diagram of housing construction actitivty is shown in Fig.3.
The conceptional diagram of the entire HOPIR system aided by computer is shown in Fig.4.
The storage space sub-system of HOPIR was first implemented in 1983, and has been in operation since then. This subsystem co-ordinates panel manufacturing, transportation and actual assembly on a real time interactive basis. The conceptual diagram of the "storage-space" subsystem is shown in Figure 5.

f/ System outputs produced at the present period:

On the ES-1022 machine: Quotation budgeting /on a computer lease basis at the Szeged Computer Centre of ÉGSZI/

On the ES-1020 machine: All processings of the "DÉLÉP INFORM" system:

- resource allocation against executive organizations
- construction site assignment schedules with resource allocation
- finished work accounting at an analytical, site-by-site as well as at an integrated level, statistical sheets
- account liquiadtion, cost monitoring
- material consupmtion accounting
- labour cost accounting
- equipment rental fees accounting
- output and revenue accounting

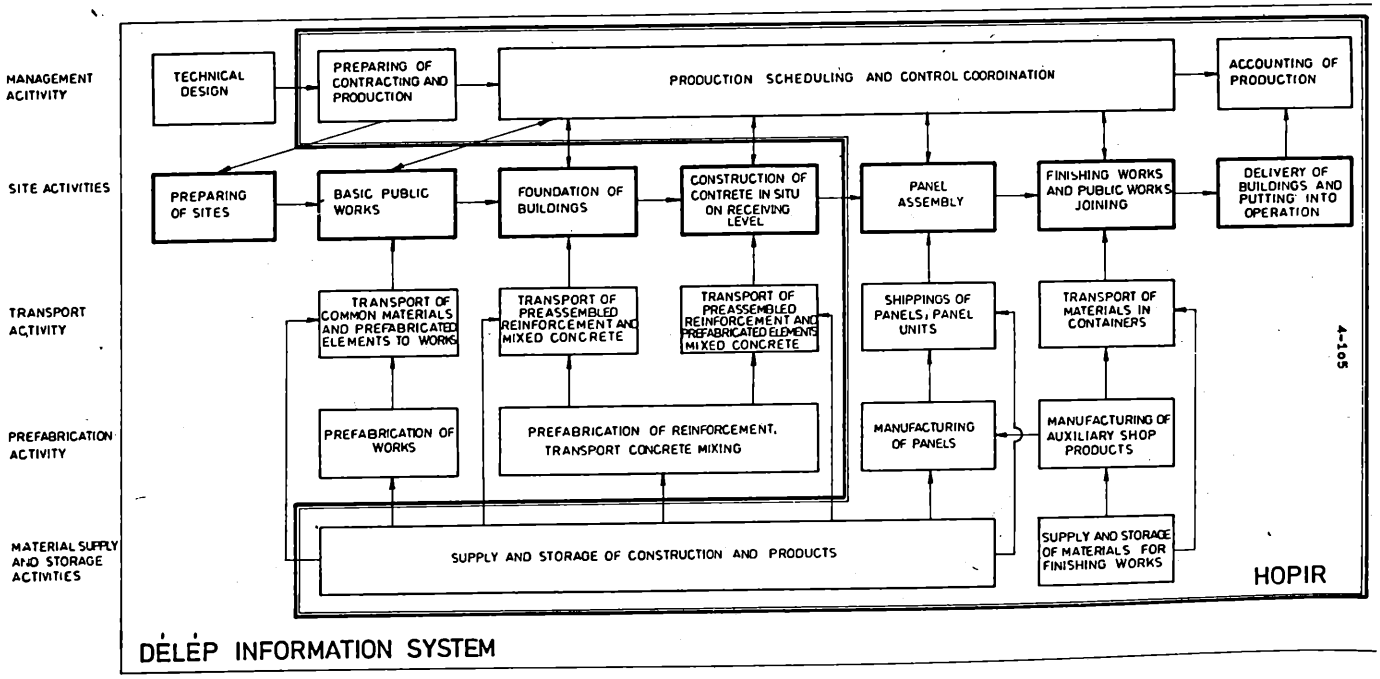


FIGURE 3: LOGICAL DIAGRAM OF HOUSING CONSTRUCTION ACTIVITY
(ZILAHY J.P. 1982.)

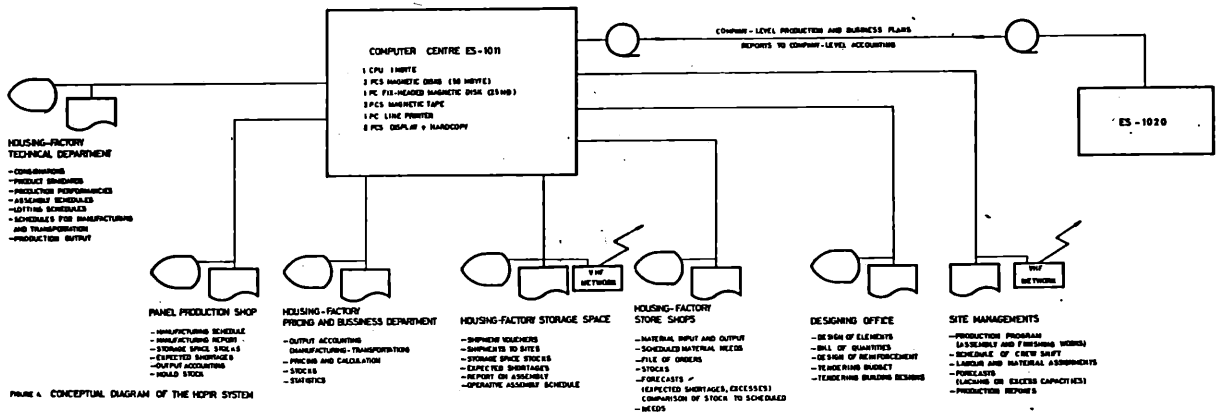
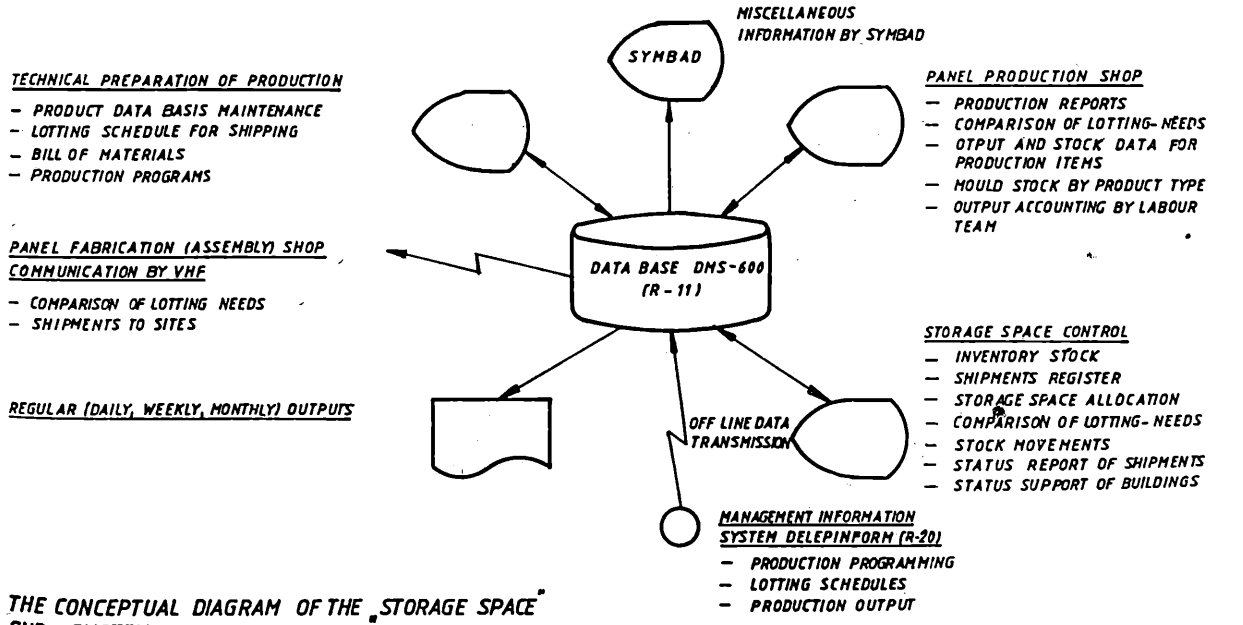


FIGURE 4. CONCEPTUAL DIAGRAM OF THE HOPIR SYSTEM

FIGURE 4. CONCEPTUAL DIAGRAM OF THE HOPIR SYSTEM
(ZELANY J.P. 1982)
(TOBIAS L.E. 1982)

FIGURE NO.5.



THE CONCEPTUAL DIAGRAM OF THE „STORAGE SPACE" SUB - SYSTEM

REG.: T.Z. TORNYI - DR. VARGA
1984.

On the ES-1011 machine: all the processings of the HOPIR system:

- storage space production control,
- programming of on-site masonry and finishing jobs, process balancing and control
- technical preparation of production
- fabrication and production control
- material storage control, container transportation control

On the Commodore-64 personal computer:

- technical and design applications /e.g.: concrete mixing formulas, dimensions of reinforced concrete support elements, etc./
- financial applications /e.g.: financial registers, forecasting follow-up evaluations, statistical computations etc./
- tender evaluation computations /e.g.: preparation of competitive quotations, quotation data base management etc./
- administrative applications /e.g.: order entry, target date registration et./

The communication between the various types of machines is ensured by the use of magnetic tapes.

4. Conclusions and Directions of development

The continued recession in the construction market compels the construction companies to develop an increasingly competitive posture. The pricing system, which has hitherto been based upon nationally binding costing standars is being gradually deregulated and business depends on competitive prices adjusted

to production costs. The reduction of the production costs and the reliable and frequent analysis of the market situation therefore became a key to success.

In the application of computer technology beyond the improving hardware and software capabilities, the human factor is seen as playing an increasingly important role. People must be prepared and trained for participating in an active and creative way in computer aided management and control systems. They must get used to the fact that their work becomes more personal, the structure of inter-relationhips more democratic; yet, at the same time what they do will become more easily measured and controlled. Beyond formal training at schools, and specialized training after graduation, it is important to provide an opportunity for on-site, "live" field training, /or application training/ where the entire range of psychological and sociological motivating means must be employed.
The application and rapid spread of personal computers may lead to the integration of certain responsibilities and functions /site technician, material controller, performance standard applicator/, and it may allow for local data processing and data transmission. The present administration oriented management structure of the company may be transformed into a project and profit commission oriented structure, changing the nature and function of data processing routines and tasks. In certain areas, the present processing routine will have to be substantially expanded. Among these are: quotation preparation, budget forecasting and computation, market analysis, market research, cost accounting and cost analysis. In all probability, the role of the small computers will be strengthened as opposed to main frame computers.

Szeged, September 1984.

References:

/1/ Zilahy, J.P.: Korszerü technológiával dolgozó vállalatok belső irányitási rendszere
Management System of Companies using developed construction technologies
/A paper for ÉTE Conference, Budapest 1977./

/2/ Sipos M. - Tornyi L.Z. - Zilahy J.P.:
Interactive Production Management System at the Housing Factory in Szeged
/A paper for the EURO-IFIP Conference in London, 1979./

/3/ Zilahy J.P.: A házgyári panelos épités számitógéppel segitett operativ irányitása
Aided Operative Production Control of Housing Construction Using Pre-Fabricated Elements
/Magyar Épitőipar 1982/7-8 szám/

/4/ Tornyi L.Z. - dr.Varga I.:
A házgyári tárolótéri irányitási rendszer kialakitása
Space Control System at the Housing Factory of DÉLÉP
/ÉGSZI Gyorsjelentés 1983/20-21 szám/

/5/ Virágh T.: The Development of the Production Management System Based on a Special Data Base in DELEP
/A paper for IKM Conference in Weimar, 1984./

Appendix No.1.

Technical configuration avaitable at the present time at the company

Configuration of Computer Type R-11

Hardware configuration:

CPU 1 Mbyte

Magnetic disks /50 Mbyte/	3 pcs
Fix-headed disk /2,5 Mbyte/	1 pc
Magnetic tape units	2 pcs
Display terminals	3 pcs
Hardcopies /80 char./	5 pcs
Line printer /132 char./	1 pc

Software configuration:

MTM-2 monitor

DMS 600 Data Base Handling System /two systems may operate simultaneously/

Configuration of computer type R-20

Hardware configuration:

CPU 256 Kbyte

Card Readers	2 pcs
Line printers /144 char./	2 pcs
Magnetic tape units	5 pcs
Magnetic tape units /7,25 Mbyte/	8 pcs
Magnetic tape /type/ data Recording posts /off line/	5 pcs

Main configuration of the PC's

PC type Commodore-64:

CPU 64 Kbyte

1 display + console

2 floppy disks /5" single page/

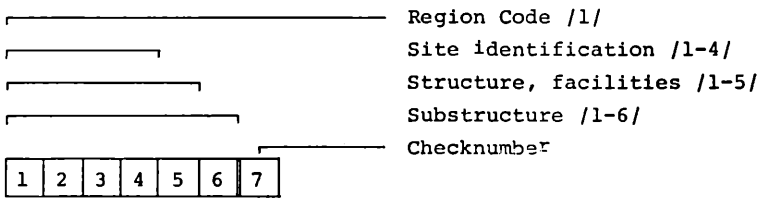
1 hard copy /80 char./

Appendix No 2.

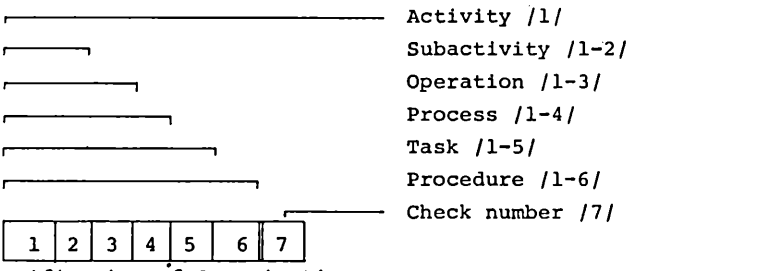
The Code Systems used in the "DELEP INFORM" and "HOPIR"

1. Identification of Construction Projects

1.1 Identification of sites

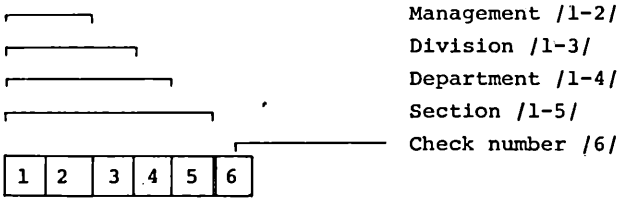


1.2 Identification of Operative Task

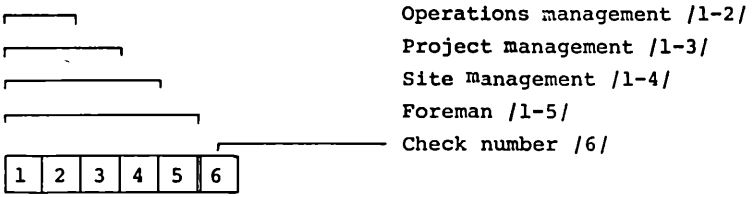


2. Identification of Organisations

2.1 Central Organisations

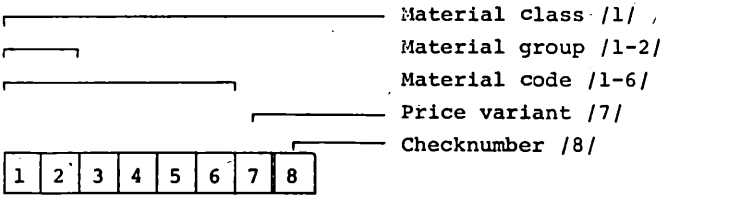


2.2 Construction and Supply Organisations

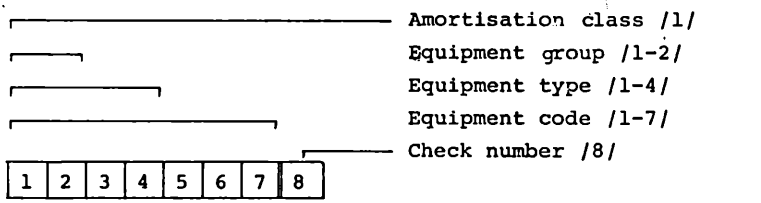


3. Identification of Resources

3.1 Materials



3.2 Construction machines



The used codes are numerical.